Fall 2013

Resource-Aware Distributed Particle Filtering for Cluster-Based Object Tracking in Wireless Camera Networks

Kihyun Hong
Purdue University

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Kihyun Hong

Entitled
Resource-Aware Distributed Particle Filtering for Cluster-Based Object Tracking in Wireless Camera Networks

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

JOHNNY PARK, Co-Chair
Chair

HONG Z. TAN, Co-Chair

HENRY MEDEIROS

MIREILLE BOUTIN

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): JOHNNY PARK, Co-Chair

Approved by: M. R. Melloch 10/09/2013
Head of the Graduate Program Date

RESOURCE-AWARE DISTRIBUTED PARTICLE FILTERING FOR CLUSTER-BASED OBJECT TRACKING IN WIRELESS CAMERA NETWORKS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Kihyun Hong

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2013

Purdue University

West Lafayette, Indiana

ACKNOWLEDGMENTS

I have received great support and encouragement from many people to complete this dissertation. I would like to thank Prof. Johnny Park for his guidance, understanding, and patience during my graduate studies at Purdue; Prof. Hong Z. Tan for encouraging me in hard times; and Dr. Henry Medeiros for sharing his knowledge and giving kind guidance as I completed this dissertation.

I would also like to thank all the members of the Robot Vision Lab, who have shared their friendship; Paul for his invaluable help with implementations of our work on a real wireless camera network; and Dave and Kyuseo for sharing their time in many discussions so that I could learn various approaches in my studies. Finally, I want to thank my wife and my family for all their love and support.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

K. Hong Ph.D., Purdue University, December 2013. Resource-Aware Distributed Particle Filtering for Cluster-Based Object Tracking in Wireless Camera Networks. Major Professors: Johnny Park and Hong Z. Tan.

The proliferation of miniaturized low-power computing devices, advances in wireless communications, and the availability of inexpensive imaging sensors have enabled the development of wireless camera networks (WCN). In this dissertation, we consider the problem of real-time object tracking with a WCN. Existing object tracking methods designed for multi-camera systems do not take into account the unique constraints of WCNs. Specifically, an effective object tracking system for WCNs must anticipate unreliable network communication, limited memory, and limited computational power in each camera node. In particular, unreliable communication degrades the quality of the visual information shared by the cameras, which ultimately degrades the tracking performance in the network. We present a novel resource-aware framework for the implementation of distributed particle filters in resource-constrained WCNs. Our method focuses on the effects of communication failures on object tracking performance by adjusting the amount of data packets generated and transmitted by the cameras according to the network conditions. We demonstrate the performance of the proposed framework using three different mechanisms to share the particle information among nodes: synchronized particles, Gaussian mixture models, and Parzen windows. We show that all three approaches benefit from the proposed resource-aware mechanism in terms of tracking accuracy or energy efficiency.

1. INTRODUCTION

Object tracking is one of the most fundamental, and at the same time, one of the most challenging problems in computer vision that is pertinent to various applications, such as automated surveillance, human-computer interaction, traffic monitoring, video indexing, etc. It is therefore not surprising that, even after decades of research, object tracking still remains one of the most actively researched topics in the computer vision community. The main bottleneck of developing object tracking algorithms is that a camera may have insufficient target information caused by object occlusion and by the indistinct nature of the object in front of cluttered backgrounds. Recent studies of object tracking methods using multiple cameras [1–5] rely on the concept of collaborative tracking of a moving target to overcome these difficulties.

In this dissertation, we focus on collaborative object tracking in networks of multiple cameras, which are suitable for large area surveillance. Along with an increasing demand on networked camera applications such as security and surveillance, the recent advances of wireless camera hardware, e.g., Imote2 [6] equipped with a camera, has brought on much interest in the development of network-based computer vision algorithms, especially object tracking algorithms for wireless camera networks (WCNs). Recently Medeiros *et al.* [7] introduced an event-driven camera clustering protocol for collaborative object tracking in camera networks. They also presented a Kalman filter based object tracking approach in a wireless network of multiple cameras using the clustering protocol in [8]. Even though the tracking algorithm was designed to run in WCNs, it did not explicitly account for the effects of communication failures, net-

work congestion, and computational load on the performance of more sophisticated collaborative tracking methods.

A number of previous works [9–11] on wireless sensor networks have presented object tracking algorithms which consider constraints related to sensor hardware such as low computational and communication capabilities. These works proposed distributed tracking methods for wireless sensor networks, which perform complicated tracking processing by spreading the computational load to local network nodes. They also presented several mechanisms to reduce the amount of data communication, e.g., incorporating data compression or parametric data representation of the communication data for the distributed tracking systems. Although these algorithms were not especially designed for WCNs, they showed tracking algorithm implementation methodologies under similar hardware limitations.

Besides the computational hardware constraints, far too little attention has been paid to communication and networking issues, such as communication channel characteristics and network traffic, in the design of object tracking methods for WCNs. These communication and networking issues are even more critical in a network of wireless cameras because vision applications tend to generate heavy network traffic and communication failures [12]. Specifically, communication failures degrade the information shared by local sensor nodes for collaborative tracking and have severe effects on object tracking performance.

This dissertation will focus on the effects of communication failures on object tracking performance and will present a communication resource-aware tracking methodology, which adjusts the amount of data packet transmission according to the network conditions. In this dissertation, valid communication packets, which are not dropped in data communications, will be considered as an available communication

resource for distributed tracking. Our approach will allow sensor nodes to consume communication energy efficiently and reduce tracking performance degradation.

1.1 Summary of Contributions

This dissertation proposes a resource-aware distributed particle filter framework for wireless camera sensor networks. We employed a widely used object tracking approach, the particle filter, using the clustering algorithm in [7] as the underlying framework for collaborative processing. We provided three mechanisms for exchanging particle information, which resulted in three different distributed particle filters for cluster-based object tracking: the synchronized particle filter, the GMM particle filter, and the Parzen particle filter. We developed a resource-aware method to improve tracking performance and energy efficiency of the distributed particle filters under lossy network environments. To summarize, the main contributions of this work are presented as follows:

1. Design of a resource-aware packet allocation mechanism for distributed particle filtering under lossy network environments.

2. Development of a framework for cluster based distributed particle filters for wireless camera networks.

3. Comparative study on the performance of distributed particle filters in lossy communication environments.

1.2 Organization

This dissertation begins by reviewing object tracking methods using a single-camera as well as multiple-cameras. It then goes on to introduce the particle filter

theory, which is the main building block of our implementation in Chapter 3. The design of the resource-aware and distributed particle filter for cluster-based wireless camera networks is described in Chapter 4. Experimental results are shown in Chapter 5. Finally, it concludes and discusses our research findings in Chapter 6.

2. OBJECT TRACKING

We define object tracking as the methodology to estimate the trajectory of a target object from observations where the observations may be acquired by a single modality or multi-modality sensors (e.g., 3D range data, sonar signals, images, etc.). All object tracking methods, regardless of sensor modality, have the common goal of estimating the most likely position of the target using the information extracted from the observation. However, the sensor modality determines the focus of the tracking algorithms. For example, tracking algorithms based on range sensor data focus on filtering noisy 3D data. Tracking algorithms based on image data, on the other hand, focus on detecting the target object in the new image frame based on the visual appearance and the position of the target object in the previous image frame. In this chapter, we will restrict our focus to image-based object tracking or visual object tracking methods.

In visual object tracking, a target object is typically represented as a set of features, e.g., pixel intensities, a color histogram [13], a histogram of gradients (HoG) [14], the scale-invariant feature transform (SIFT) [15], etc., with different advantages and disadvantages for each feature. For example, a color histogram is robust to structural changes such as pose change, partial occlusions, etc., whereas an HoG feature is suitable for describing the human appearance. In this dissertation, we will limit our discussion to the most common histogram-based object tracking algorithms, and in particular, color histograms [13, 16–20].

2.1 Single-Camera Object Tracking

In this section, we will briefly review the basic principles of single-camera object tracking methods. Various types of object tracking methods such as Meanshift [13], Camshift [21], and particle filters [16–18] have been proposed. We can categorize these object tracking methods into two types: unimodal and multimodal tracking.

In unimodal tracking [13, 22], the probability distribution that represents the position of the target object is assumed to have a single mode. Theoretically, unimodal tracking computes the MAP (maximum a posterior) estimates as the current position of the object in each frame:

$$\hat{\mathbf{x}}_t = \arg\max_{\mathbf{x}_t} \left\{ p(\mathbf{x}_t|\mathbf{y}_t) \right\}, \tag{2.1}$$

where $\mathbf{x}_t$ is the object position in the spatial domain and $\mathbf{y}_t$ is the observation at time t.

If the prior probability of the object position $p(\mathbf{x}_t)$ follows an additive white Gaussian noise (AWGN) model, the target position can be found using the minimum difference criterion between the reference model and the observation on the current frame:

$$\hat{\mathbf{x}}_t = \arg\min_{\mathbf{x}_t} \left\{ ||f(\mathbf{x}_t) - \mathbf{y}_t||_2 \right\}, \tag{2.2}$$

where $f(\mathbf{x}_t)$ is the feature vector at $\mathbf{x}_t$, and $\mathbf{y}_t$ is the reference feature model. Sum of squared difference (SSD), Meanshift, and Camshift are some of the most popular unimodal tracking methods that essentially try to find the solution of Eq. (2.2). These methods are in general easy to implement and known to be computationally efficient. However, due to the unimodality assumption, they are susceptible to get trapped in a local minimum especially when the target object is partially occluded or when there

exists a visually similar object near the target object. Some researchers have employed the Kalman filter [13] for object tracking. In this approach, the Kalman recursive formation, which assumes Gaussian noise process and linear recursive dynamics, uses the estimated target positions produced by color-based trackers as noisy observations and performs a temporal filtering of these estimated positions. However, even with the help of Kalman filtering, unimodal trackers often perform poorly when the target object undergoes occlusion or when there are visually similar non-target objects near the target object.

Multimodal tracking approaches allow multiple modes in the posterior probability distribution of the target position, thus they are in general more robust to occlusions and competing non-target objects. Particle filtering is the most commonly used technique in multimodal-based tracking [16,18]. The particle filter approximates multiple modes using a Monte Carlo (MC) method and the tracked position is obtained by the minimum mean squared error (MMSE) estimate:

$$\hat{\mathbf{x}}_t = E(\mathbf{X}_t|\mathbf{y}_{0:t}). \tag{2.3}$$

We will provide a detailed review of the tracking methods based on particle filtering in Chapter 3.

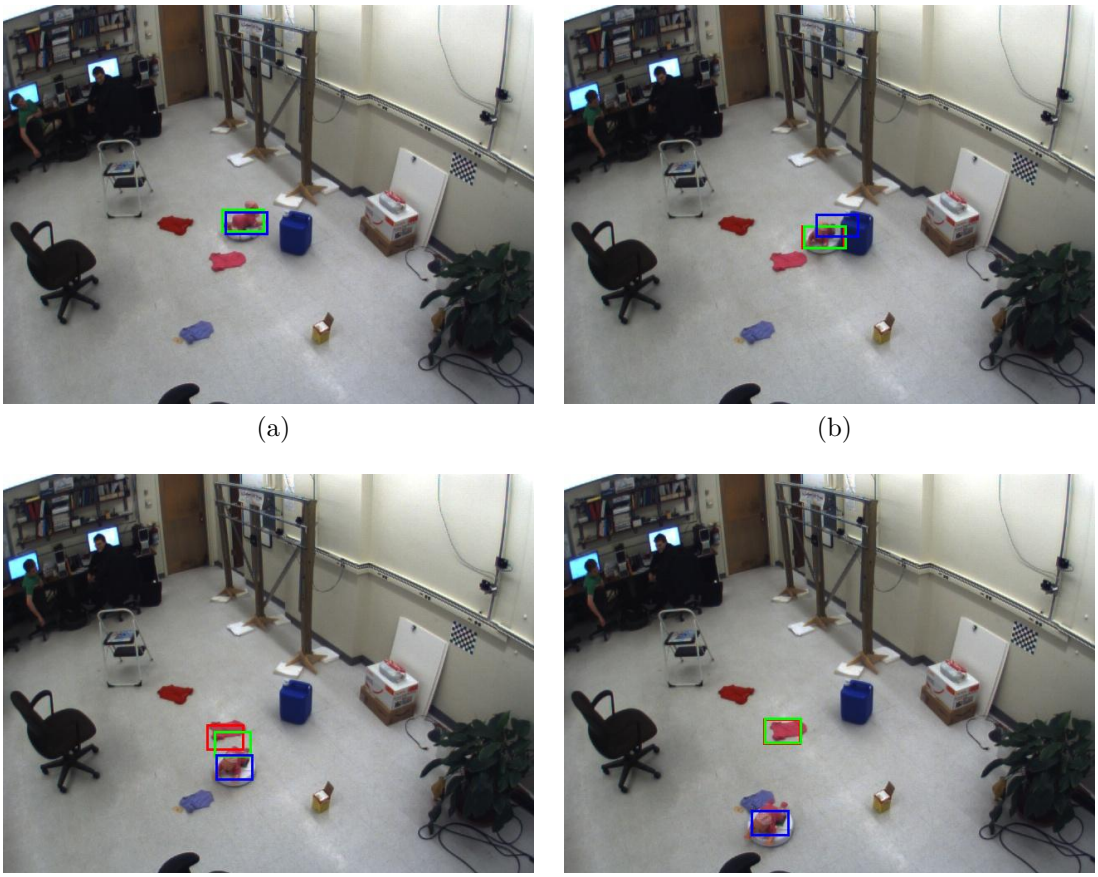
Fig. 2.1 shows some examples of unimodal and multimodal trackers. In these examples, all trackers have been programmed to track the pink baby doll. The green box represents the estimated position by a unimodal tracker using Meanshift. The red box represents the estimated position by the same tracker but with Kalman filtering, and finally the blue box is the estimated position of the multimodal tracker with particle filtering. This example illustrates how the unimodal trackers can be easily "trapped" onto a visually similar non-target object whereas the multimodal tracker using particle filtering shows robustness against the background distractions

and competing non-target objects. Fig. 2.2 shows the probability distribution of the position of the target object in each frame. One can observe the highly non-Gaussian and multimodal characteristics of the distributions, which explain the better tracking performance by the multimodal-based tracker.

2.2 Multi-Camera Object Tracking

Object tracking using a single camera, because of the camera's limited viewing area and direction, cannot track objects in a large area and is prone to lose track when the object is fully occluded. If multiple cameras, each with different viewing area and direction, collaborate on object tracking, one can expect to have a larger tracking coverage with more robustness against occlusion [1–5]. Fig. 2.3 illustrates the advantage of object tracking using multiple cameras. In this example, a single-camera tracker (whose tracking result is represented by the red boxes in the images) loses the track of the target object, i.e., the pink baby doll, as the object gets partially occluded. However, a multi-camera based tracker whose tracking result is represented by the white boxes in the images is able to continually track the object with help of other cameras (in this case, three other cameras). More importantly, when multiple cameras are used in object tracking, we have the ability to compute the 3D coordinates of the target object using multi-view geometry.

To gain the benefits of using multiple cameras in object tracking applications, various approaches have been proposed. They are mostly categorized into direct information fusion and probabilistic information fusion. In the direct information approach [3–5], 2D trackers (single view based trackers) are executed individually and the final 3D track is estimated by combining the 2D tracks with geometric or appearance constraints. Let us say $\hat{\mathbf{x}}_{i,t}$ is the 2D track result of the i^{th} camera and



(a) (b)

(c) (d)

Figure 2.1.: Examples of unimodal and multimodal trackers. All trackers have been programmed to track the pink baby doll. The green box represents the estimated position by a unimodal tracker using Meanshift and the red box represents the estimated position by the same tracker but with Kalman filtering. Finally the blue box is the estimated position of the multimodal tracker with particle filtering. (a) frame 35, (b) frame 62, (c) frame 83, and (d) frame 97.

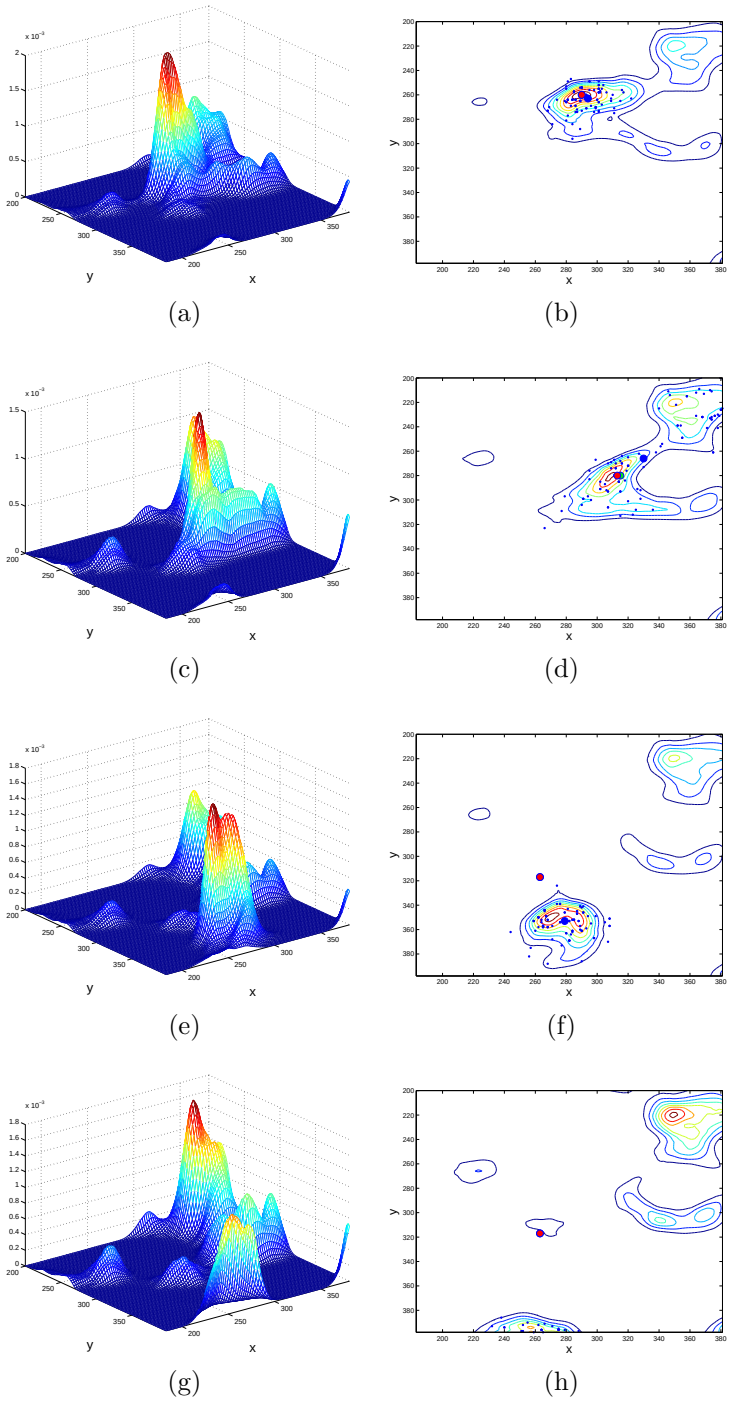
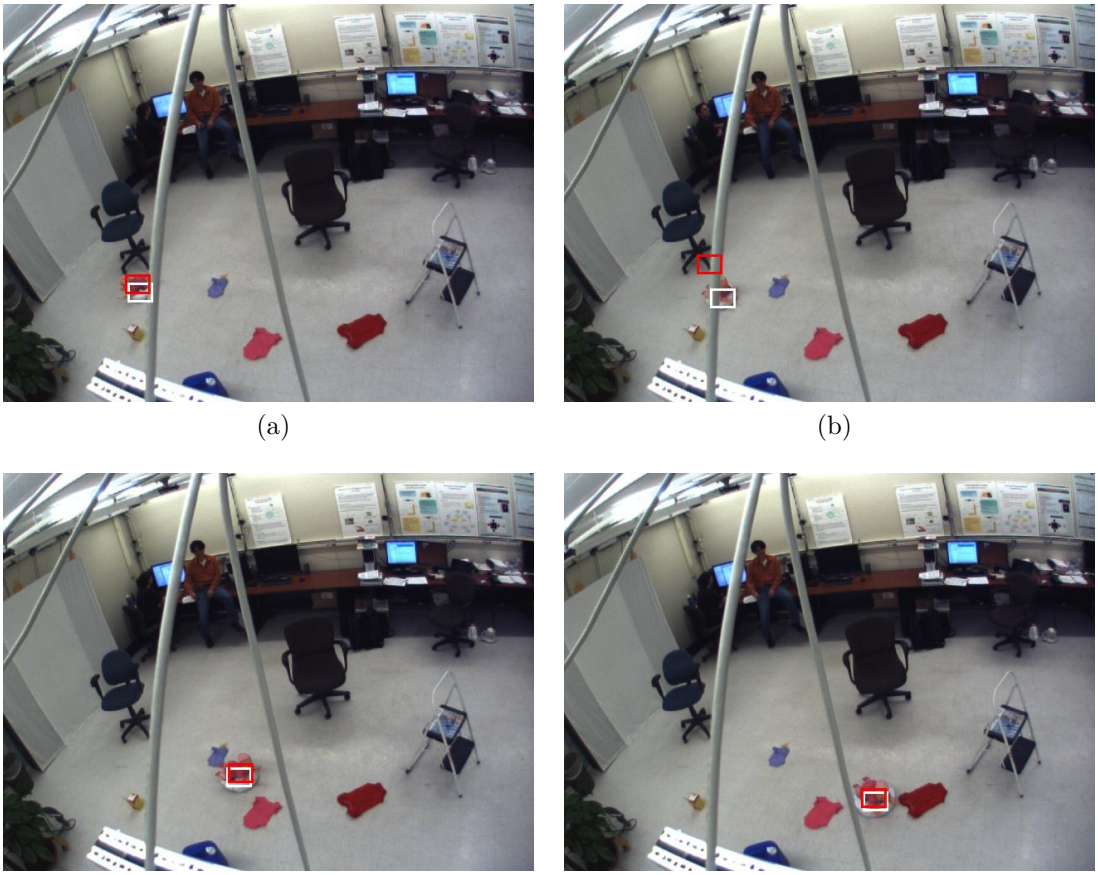


Figure 2.2.: Object posterior probabilities. (a) and (b): frame 35; (c) and (d): frame 62; (e) and (f): frame 83; and (g) and (h): frame 97. Green, red, and blue circles represent the tracking results by Meanshift, Meanshift with Kalman filtering, and multimodal with particle filtering, respectively. The small blue dots indicate the particle samples used to estimate the posterior probabilities.



(a) (b)

(c) (d)

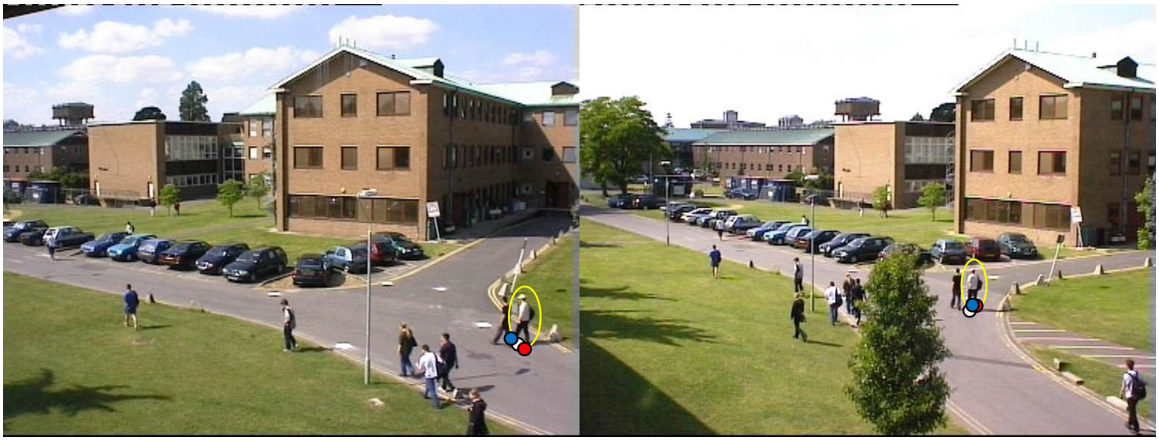
Figure 2.3.: Tracking results by a single camera and multiple cameras; Red and white boxes represent single camera based and multiple camera (4 cameras) based particle filter tracking, respectively. (a) frame 2, (b) frame 42, (c) frame 64, and (d) frame 88 in Lab sequence.

$g(\cdot)$ is an information fusion function. Then, the multi-camera fusion approach can generally be expressed as

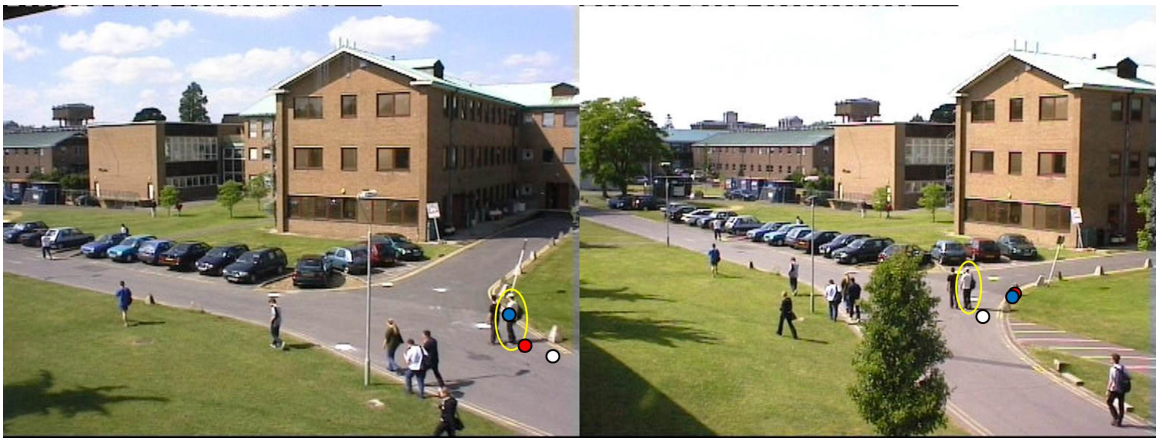
$$\hat{\mathbf{x}}_t = g(\hat{\mathbf{x}}_{1,t}, \hat{\mathbf{x}}_{2,t}, \cdots, \hat{\mathbf{x}}_{N,t}), \tag{2.4}$$

where N is the number of cameras. Typically, the information fusion function g uses a selection or average process; the function $g(\cdot)$ selects the local estimate (the single view based track) which has the minimum estimate error, or it averages the local estimates with filtering or weighted summation. Nummiaro *et al.* [3] presented a selection based multi-view particle filter. Each camera tracking unit runs independently and camera collaboration is done by selection of the best camera tracking result, which shows the appearance probability among all camera tracks. Also, multiple camera information such as epipolar geometry is used for target initialization and for the detection of loss of tracking that may call for reinitialization. In [8], Kalman filter based multi-camera collaboration was proposed. In this application, a 2D estimate of each view is transformed to the world coordinate plane by a homography. The transformed 3D track of each view (individual object position estimates) is sequentially applied to a single Kalman filter. The estimated 3D object track by the Kalman filter is a weighted average of the available camera unit tracks since Kalman filtering is a linear process. However, these approaches do not realize the full gain available from multiple observations, since inherently these approaches are based on 2D tracking separately in the various camera units, which have limited performance as mentioned previously. Fig. 2.4 shows a Kalman based multi-camera tracking example. In this example, when one camera unit loses object track (or has very noisy track), the Kalman filter based fusion is not able to estimate the object position correctly.

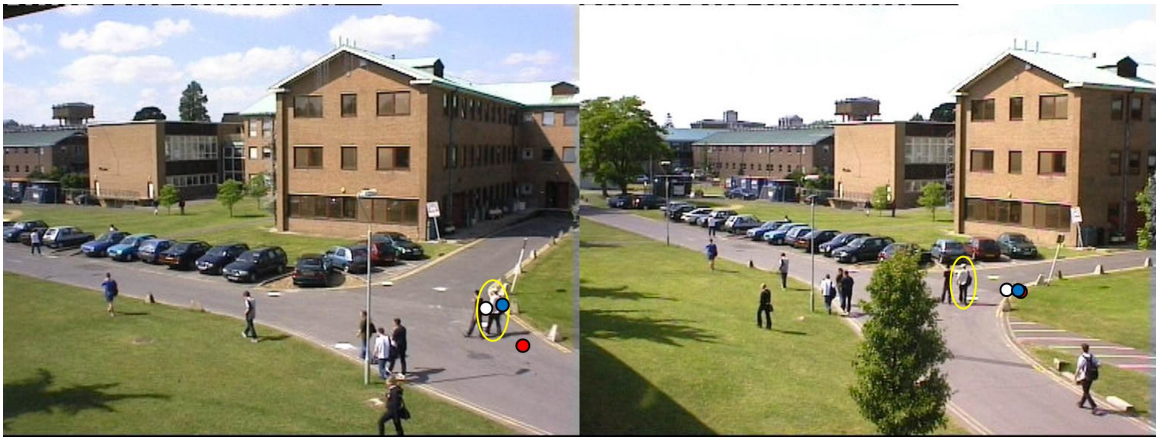
On the other hand, probabilistic information approaches [1, 2] consider forming the posterior probability given all camera observations, $p(\mathbf{x}_t|\mathbf{y}_{1,0:t}, \mathbf{y}_{2,0:t}, \cdots, \mathbf{y}_{N,0:t})$,



(a)



(b)



(c)

Figure 2.4.: Kalman filter based multiple camera tracking results; yellow circle shows the target object and red, blue, and white dots represent independent Meanshift tracking results, the average of Meanshift tracking, and Kalman combined results. (a) frame 1, (b) frame 12, and (c) frame 36 in PETS sequence.

instead of combining individual estimates. Then, object tracking can be formulated as

$$\hat{\mathbf{x}}_t = \arg\max_{\mathbf{x}} \left\{ p(\mathbf{x}_t | \mathbf{y}_{1,0:t}, \mathbf{y}_{2,0:t}, \cdots, \mathbf{y}_{N,0:t}) \right\}, \tag{2.5}$$

or

$$\hat{\mathbf{x}}_t = E(\mathbf{X}_t | \mathbf{y}_{1,0:t}, \mathbf{y}_{2,0:t}, \cdots, \mathbf{y}_{N,0:t}). \tag{2.6}$$

If the probability distribution is correct, theoretically the multiple observation based estimates in Eqs. (2.5) and (2.6) have equal or less error than the single observation based estimates in Eqs. (2.1) and (2.3). As long as the observations of the cameras are not identical, the estimates based on multiple observations reduce the error between the true object position and the estimated position. Generic extensions of the particle filter for object tracking to multiple camera applications can be found in [1] and [2].

Through some examples, we compare direct and probabilistic information fusions. Let us say the i^{th} local camera tracking estimate is $\hat{\mathbf{x}}_{i,t} = E(\mathbf{X}_t | \mathbf{y}_{i,0:t})$. Then the direct information fusion approach can be formulated as

$$\hat{\mathbf{x}}_t = g(E(\mathbf{X}_t | \mathbf{y}_{1,0:t,}), E(\mathbf{X}_t | \mathbf{y}_{2,0:t}), \cdots, E(\mathbf{X}_t | \mathbf{y}_{N,0:t})). \tag{2.7}$$

If we choose g as a linear function, then the estimate is just a weighted average of the local estimates which is

$$\hat{\mathbf{x}}_t = w_1 E(\mathbf{X}_t | \mathbf{y}_{1,0:t}) + w_2 E(\mathbf{X}_t | \mathbf{y}_{2,0:t}) + \cdots + w_N E(\mathbf{X}_t | \mathbf{y}_{N,0:t}), \tag{2.8}$$

where in a selection process, the weights w_n are adaptively set to one for the best result and zeros for the others. This approach only guarantees local optimality (i.e., each camera unit) not global optimality; if the global optimum is not located in the span of

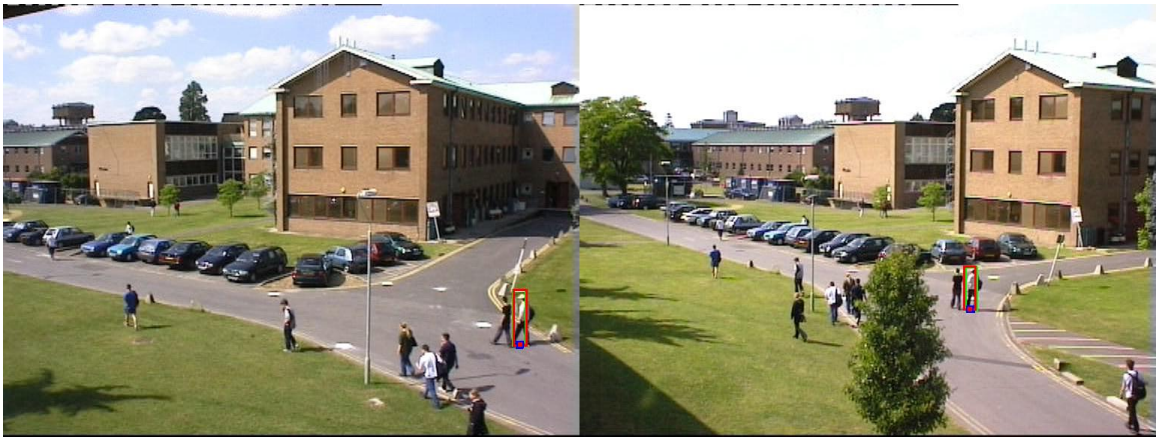
the local optima, any weighted summation of local optima can not be the same as the global optimum. On the other hand, the multiple observation based MMSE estimates in Eq. (2.6) can minimize the probabilistic uncertainty and also can track the object in the world coordinate more accurately by forming a global (joint) posterior probability. Figs. 2.5 and 2.6 show the comparison between a global MMSE estimate and a direct information fusion based particle filters. For the local fusion example, the weights of Eq. (2.8) are equally assigned. This local fusion approach is vulnerable to outliers since it is an average process. Figs. 2.8 and 2.10 show single camera based tracking results and global estimation result and Figs. 2.9 and 2.11 depict the ellipsoids of constant probability when the particle is modeled using a Gaussian distribution (to show a graphical interpretation even though the distributions are not Gaussian). This example shows that global probability derived from probabilistic information fusion has smaller variance (uncertainty) than independent local probabilities without any help of other observations.

In summary, the performance of probabilistic information fusion based trackers, especially probabilistic information fusion based on particle filter, is superior compared to the direct information fusion approaches.

(a)

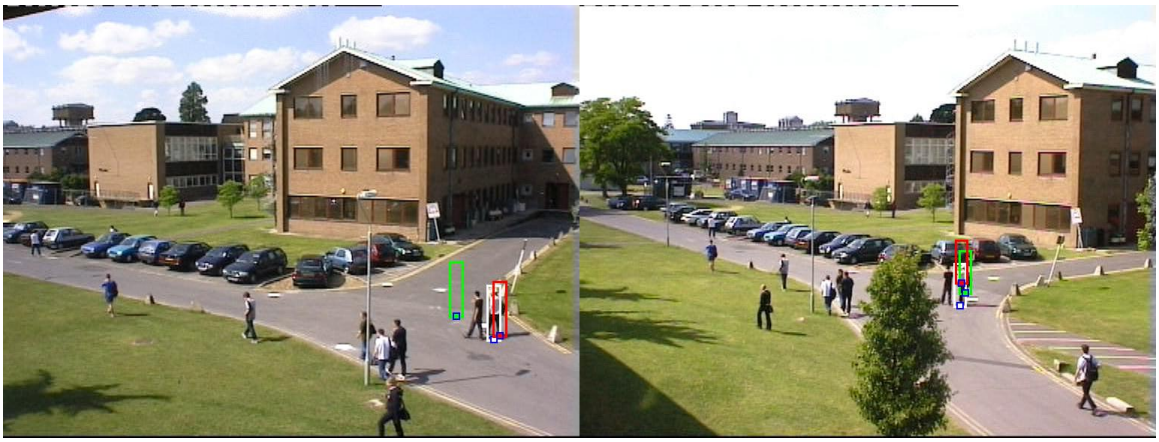
(b)

Figure 2.5.: Comparison between local and global particle filter trackers in the Lab sequence. The red boxes represent the tracking results by independent local particle filters, the green boxes represent the average of the local particle filters, and the white boxes the global particle filter. (a) frame 42 and (b) frame 86 in Lab sequence.



(a)

(b)



(c)

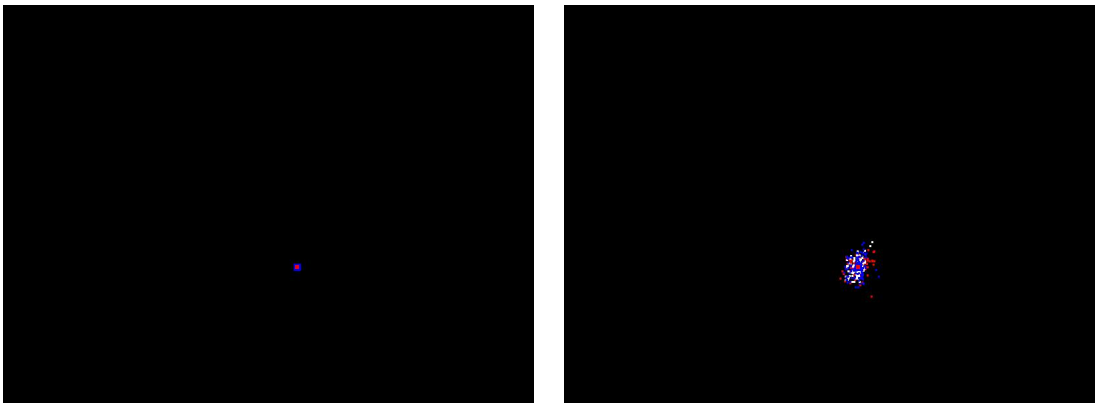
Figure 2.6.: Comparison between local and global particle filter trackers in the PETS sequence. The red boxes represent the tracking results by independent local particle filters, the green boxes represent the average of the local particle filters, and the white boxes the global particle filter. (a) frame 1, (b) frame 12, and (c) frame 36 in PETS sequence.

(a)

(b)

(c)

Figure 2.7.: Single independent and multiple observation (global) particles; red dots and blue dots are single camera based particles and white dots are global particles. (a) frame 1, (b) frame 12, and (c) frame 36 in PETS sequence.



(a) (b)

Figure 2.8.: Multiple independent and one global estimations at frame 1; (a) particle filter tracking estimation and (b) resampled posterior particles in world coordinates. (a) the single independent particle estimations, the average of the single particle estimation, and the global estimation are marked by red dots, green dot and white dot, respectively. (b) blue and red dots represent each camera particles and white dots are global posterior particles.

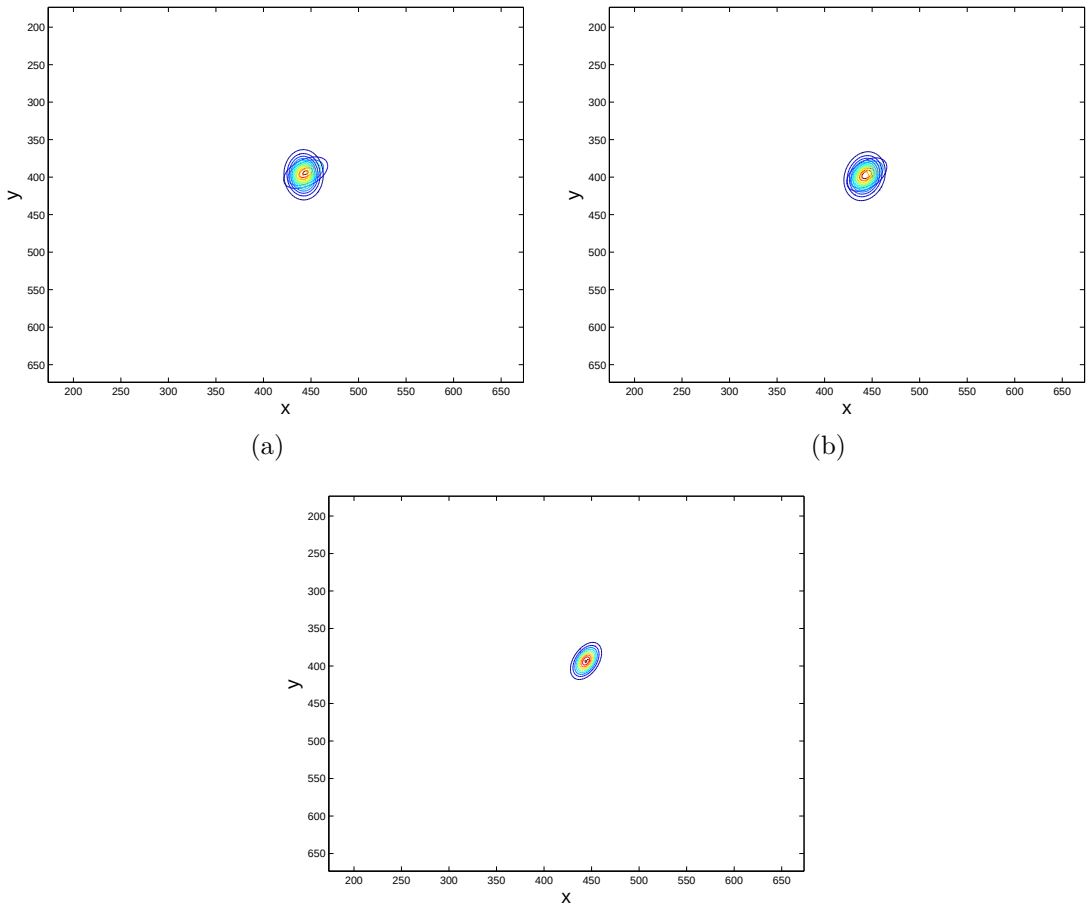
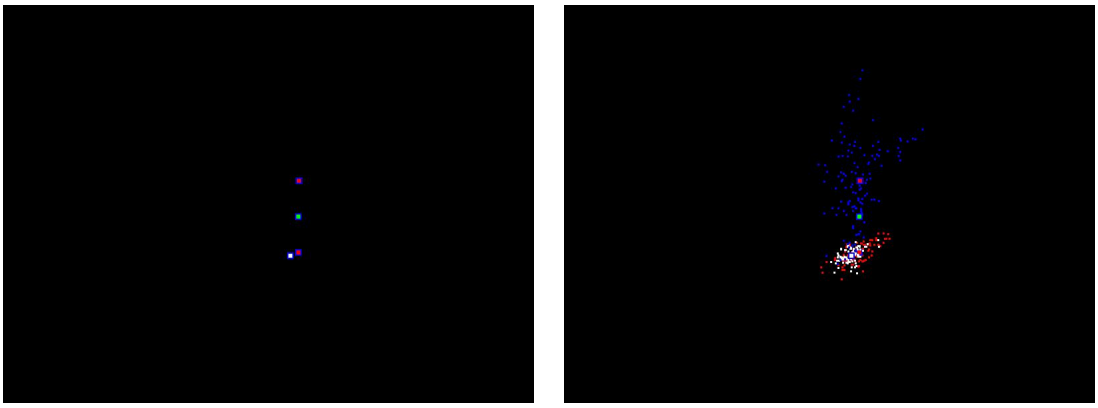


Figure 2.9.: Particle filter posterior probabilities at frame 1; (a) two independent single posterior probabilities, (b) two local posterior probabilities, and (c) global posterior probability in world coordinates.



(a) (b)

Figure 2.10.: Multiple independent and one global estimations at PETS sequence frame 36; (a) particle filter tracking estimation and (b) resampled posterior particles in world coordinates. (a) the single independent particle estimations, the average of the single particle estimation, and the global estimation are marked by red dots, green dot and white dot, respectively. (b) blue and red dots represent each camera particles and white dots are global posterior particles.

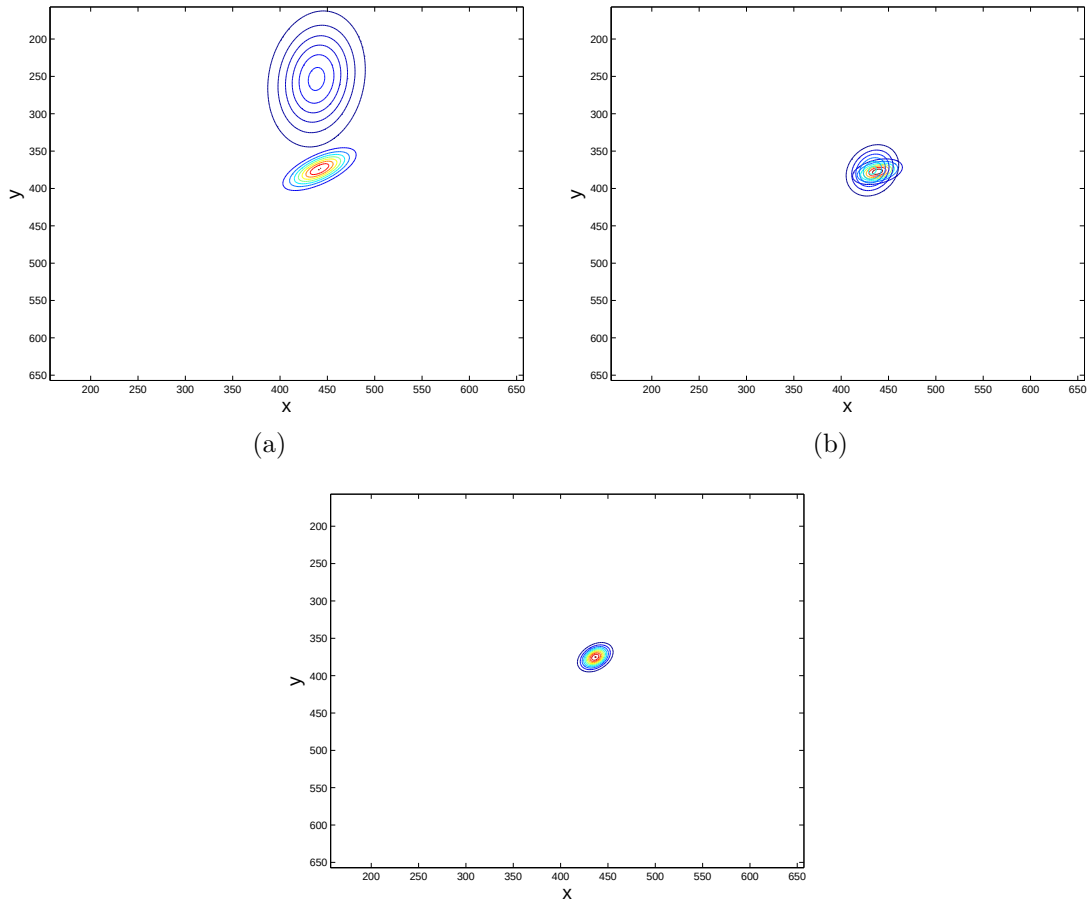


Figure 2.11.: Particle filter posterior probabilities at PETS sequence frame 36; (a) two independent single posterior probabilities, (b) two local posterior probabilities, and (c) global posterior probability in world coordinates.

3. DISTRIBUTED PARTICLE FILTER

In object tracking, as mentioned in the previous chapter, we can improve the tracking by accommodating multiple measurements or observations, which is one of the main purposes of sensor networks. Before we consider distributed particle filtering methods, we first review the basic particle filter theory in Section 3.1 and particle filtering with multiple observations in Section 3.2. Then, we move to distributed particle filters for sensor networks in Section 3.3.

3.1 Particle Filter

Let us denote the state random process at time t as $\mathbf{X}_t$ and its realization as $\mathbf{x}_t$. The observation and its value at time t will be denoted $\mathbf{Y}_t$ and $\mathbf{y}_t$, respectively. Then, a dynamic system is defined by a state process model:

$$\mathbf{X}_{t+1} = f_t(\mathbf{X}_t, \mathbf{U}_t), \tag{3.1}$$

and an observation model,

$$\mathbf{Y}_t = h_t(\mathbf{X}_t, \mathbf{V}_t), \tag{3.2}$$

where $\mathbf{U}_t$ and $\mathbf{V}_t$ are independent white noises and independent identically distributed (IID) processes regarding t, and f_t and h_t are assumed to be known functions. In a probabilistic dynamic system model, the process equation in Eq. (3.1) and observation model in Eq. (3.2) of the state equations can be obtained from the update

(or transition) probability, $p_{\mathbf{X}_{t+1}|\mathbf{X}_t}(\mathbf{x}_{t+1}|\mathbf{x}_t)$, and observation likelihood probability, $p_{\mathbf{Y}_t|\mathbf{X}_t}(\mathbf{y}_t|\mathbf{x}_t)$:

$$\mathbf{X}_{t+1} = f_t(\mathbf{X}_t, \mathbf{U}_t) \Leftrightarrow p_{\mathbf{X}_{t+1}|\mathbf{X}_t}(\mathbf{x}_{t+1}|\mathbf{x}_t), \tag{3.3}$$

$$\mathbf{Y}_t = h_t(\mathbf{X}_t, \mathbf{V}_t) \Leftrightarrow p_{\mathbf{Y}_t|\mathbf{X}_t}(\mathbf{y}_t|\mathbf{x}_t). \tag{3.4}$$

Typically object tracking using a probabilistic dynamic model consists of the following two problems [23]:

- Compute the posterior state probability at each t, $p_{\mathbf{X}_t|\mathbf{Y}_{0:t}}(\mathbf{x}_t|\mathbf{y}_{0:t})$, given all observations up until time t, $\mathbf{y}_{0:t} = (\mathbf{y}_0, \cdots, \mathbf{y}_t)$.
- Estimate a functional output of the state at t, $g(\mathbf{X}_t)$. For example, the minimum mean square error (MMSE) estimate $E(g(\mathbf{X}_t)|\mathbf{y}_{0:t})$.

For a linear dynamic system with Gaussian noise process, the Kalman filter has been commonly used to solve the probabilistic dynamic equation. However, for nonlinear systems which may have non Gaussian process, the Kalman filter approach is not suitable since it is a linear estimator. On the other hand the particle filter is able to handle the probabilistic dynamic state models which are nonlinear/non Gaussian.

In the particle filter method, the posterior probability $p_{\mathbf{X}_t|\mathbf{Y}_{0:t}}(\mathbf{x}_t|\mathbf{y}_{0:t})$ is formulated using the recursive Bayesian inference with the state process and observation models. The MMSE estimate $E(g(\mathbf{X}_t)|\mathbf{y}_{0:t})$, which is generally a nonlinear estimate, is obtained by a Monte Carlo (MC) approximation with the computed posterior probability. If we apply a linear MMSE (LMMSE) estimate to the above problem, the solution is equivalent to the Kalman filter.

From now on, for the sake of notational simplicity, we will omit random variable subscripts in the probability density function (PDF) expressions, e.g., $p_{\mathbf{X}_t}(\mathbf{x}_t) \Rightarrow p(\mathbf{x}_t)$.

In the rest of this chapter, we will review the particle filter theory as a sequential Monte Carlo method. Then, its extension to the multiple observation case will be presented. Finally, we will discuss distributed particle filtering implementations.

3.1.1 Sequential Monte Carlo Methods

In this section, we will first briefly review the MC simulation concept which is a basic building block of the particle filter. This will lead into the topic of particle filter, which is simply a recursive Bayesian inference approach using sequential MC estimations. Then, we will introduce the multiple observation based particle filter that is the framework of centralized and distributed particle filter implementations.

3.1.1.1 Monte Carlo Simulation

Suppose a set of K samples $\{x^{(k)}\}_{k=1}^{K}$ are drawn from a PDF $p(x)$:

$$\{x^{(k)}\}_{k=1}^{K} \sim p(x). \tag{3.5}$$

Then, the PDF can be approximated by the samples if K is large enough:

$$p(x) \approx \frac{1}{Z} \sum_{k=1}^{K} \delta(x - x^{(k)}), \tag{3.6}$$

where Z is a normalization constant, in this case $Z = K$, and $\delta(\cdot)$ is the Kronecker delta function.

Now, let's consider a joint PDF consisting of two independent PDFs, $p(x) = p_1(x)p_2(x)$. Suppose we have a set of samples that are drawn from $p_2(x)$, $\{x^{(k)}\}_{k=1}^{K} \sim$

$p_2(x)$, and $p_1(x)$ is measurable at these points. Then, the joint PDF can be approximated as

$$\begin{aligned} p(x) &= p_1(x)p_2(x) && (3.7)\\ &\approx \frac{1}{Z}p_1(x)\sum_{k=1}^{K}\delta(x-x^{(k)}) && (3.8)\\ &= \frac{1}{Z}\sum_{k=1}^{K}p_1(x^{(k)})\delta(x-x^{(k)}), && (3.9) \end{aligned}$$

where $Z=\sum_{k=1}^{K}p_1(x^{(k)})$. Defining $w^{(k)}=\frac{p(x^{(k)})}{Z}$, the PDF $p(x)$ can be characterized by the *random measure* [24], $\{x^{(k)},w^{(k)}\}_{k=1}^{K}$, which is a set of support points $x^{(k)}$ and their associated weights $w^{(k)}$:

$$p(x)\approx\sum_{k=1}^{K}w^{(k)}\delta(x-x^{(k)}). \quad (3.10)$$

In MC simulation, an unknown density function is approximated by the above technique. Let us say $p(x)$ is an unknown PDF which can be measured at support points and $q(x)$ is a known PDF which can be sampled. Also, let us assume the samples drawn from $q(x)$ support the domain of $p(x)$. Let us define a function $r(x)=\frac{p(x)}{q(x)}$ which is also measurable at the support points. Then, $p(x)$ can be expressed as

$$p(x)=r(x)q(x). \quad (3.11)$$

By applying the samples of $q(x)$ to the equation, $p(x)$ can be approximated as

$$\begin{aligned} p(x) &\approx \frac{1}{Z}\sum_{k=1}^{K}r(x^{(k)})\delta(x-x^{(k)}) && (3.12)\\ &= \sum_{k=1}^{K}w^{(k)}\delta(x-x^{(k)}), && (3.13) \end{aligned}$$

where $Z = \sum_{k=1}^{K} r(x^{(k)})$ and $w^{(k)} = \frac{r(x^{(k)})}{Z}$. The unknown $p(x)$ can be described by the random measure $\{x^{(k)}, w^{(k)}\}_{k=1}^{K}$. Note that the PDF $q(x)$ which is utilized for sampling is also referred as a *proposal distribution* or *importance distribution*, and the MC method is called *importance sampling* [25]. Finally, the MC estimation is done by computing the expectation of the above discrete approximation of the distribution $p(x)$:

$$E(X) = \int x p(x) dx \tag{3.14}$$

$$\approx \int x \sum_{k=1}^{K} w^{(k)} \delta(x - x^{(k)}) dx \tag{3.15}$$

$$= \sum_{k=1}^{K} w^{(k)} x^{(k)}, \tag{3.16}$$

which is the weighted summation of the samples. MC recursive Bayesian inference, especially particle filtering, is derived from these sampling, discrete approximation, and estimation methods. In the next sub-section, we will discuss the particle filtering theory in detail.

3.1.1.2 Recursive Bayesian Inference in Monte Carlo Simulation

In this sub-section, we will study the recursive Bayesian inference to formulate the posterior probability $p(\mathbf{x}_t|\mathbf{y}_{0:t})$ in the probabilistic dynamic system. Let us assume that the current state at t in Eq. (3.3) is only statistically dependent on the previous state at $t-1$ and the observations in Eq. (3.4) are conditionally independent given a hidden state - a hidden Markov model (HMM) which is normally assumed in tracking applications. Then, the posterior probability $p(\mathbf{x}_t|\mathbf{y}_{0:t})$ is decomposed as follows:

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1}) d\mathbf{x}_{t-1}}{p(\mathbf{y}_t|\mathbf{y}_{0:t-1})}, \tag{3.17}$$

where $p(\mathbf{y}_t|\mathbf{y}_{0:t-1})$ is the normalization constant. In a probabilistic model, the equation is tackled in two steps - prediction and update:

- Prediction:

$$p(\mathbf{x}_t|\mathbf{y}_{0:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1})d\mathbf{x}_{t-1}. \tag{3.18}$$

- Update:

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{0:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{0:t-1})}. \tag{3.19}$$

In the prediction step, given the previous posterior PDF $p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1})$, we obtain the prior PDF $p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$ by the Chapman-Kolmogorov equation [26], which marginalizes the probability of reaching the state $\mathbf{x}_t$ given the previous state $\mathbf{x}_{t-1}$ and the previous observations. In the MC approach, given the random measure $\{\mathbf{x}_{t-1}^{(k)}, w_{t-1}^{(k)}\}_{k=1}^{K}$ that characterizes $p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1})$, the distribution $p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$ can be formed as

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{y}_{0:t-1}) &= \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1})d\mathbf{x}_{t-1} \\ &\propto \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) \sum_{k=1}^{K} w_{t-1}^{(k)} \delta(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^{(k)}) d\mathbf{x}_{t-1} \qquad (3.20) \\ &= \sum_{k=1}^{K} p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)}) w_{t-1}^{(k)}. \qquad (3.21) \end{aligned}$$

Therefore, if we have a set of current state samples $\{\mathbf{x}_t^{(k)}\}_{k=1}^{K}$ from a proposal distribution $q(\mathbf{x}_t)$, then the prior PDF, $p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$, can be described by the following random measure,

$$\left\{ \mathbf{x}_t^{(k)}, \frac{1}{Z} \frac{\sum_{n=1}^{K} p(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(n)}) w_{t-1}^{(n)}}{q(\mathbf{x}_t^{(k)})} \right\}_{k=1}^{K}, \tag{3.22}$$

where Z is the normalization constant. Then the update procedure will be straightforward. By applying the prior random measure in Eq. (3.22) to Eq. (3.19), we get the current state posterior PDF as

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) \approx \sum_{k=1}^{K} w_t^{(k)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(k)}), \tag{3.23}$$

where the updated weight is

$$w_t^{(k)} \propto \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(k)}) \sum_{n=1}^{K} p(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(n)}) w_{t-1}^{(n)}}{q(\mathbf{x}_t^{(k)})}. \tag{3.24}$$

Note that we commonly set the proposal density dependent on the previous state and observation which is $q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{y}_t)$, since the density supports the domain of the likelihood PDF and the samples from the importance distribution well represent the dynamic relations.

In the particle filter, the sequential importance sampling (SIS) [17, 27] draws the current state samples from the previous samples recursively:

$$\mathbf{x}_t^{(k)} \backsim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)}, \mathbf{y}_t). \tag{3.25}$$

Then, the weights in Eq. (3.24) are formed as

$$w_t^{(k)} \propto w_{t-1}^{(k)} \frac{p(\mathbf{y}_t|\mathbf{x}_t^{(k)}) p(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(k)})}{q(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(k)}, \mathbf{y}_t)}. \tag{3.26}$$

Hence, in the prediction step of the SIS particle filter, we have

$$p(\mathbf{x}_t^{(k)}|\mathbf{y}_{0:t-1}) = w_{t-1}^{(k)} \frac{p(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(k)})}{q(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(k)}, \mathbf{y}_t)}. \tag{3.27}$$

And in the update step, the likelihood weights are computed as

$$w_t^{(k)} = \frac{1}{Z} p(\mathbf{y}_t|\mathbf{x}_t^{(k)}) p(\mathbf{x}_t^{(k)}|\mathbf{y}_{0:t-1}), \tag{3.28}$$

with the following normalization,

$$Z = \sum_{k=1}^{K} w_t^{(k)}. \tag{3.29}$$

Then, we have the posterior random measure and can approximate the posterior probability:

$$\{\mathbf{x}_t^{(k)}, w_t^{(k)}\}_{k=1}^{K} \Leftrightarrow p(\mathbf{x}_t|\mathbf{y}_{0:t}), \tag{3.30}$$

and

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) \approx \sum_{k=1}^{K} w_t^{(k)} \delta(\mathbf{x}_t - \mathbf{x}_t^{(k)}). \tag{3.31}$$

From the posterior PDF, we estimate a functional output of the state at t, $g(\mathbf{X}_t)$ as

$$E(g(\mathbf{X}_t)|\mathbf{y}_{0:t}) \approx \sum_{k=1}^{K} w^{(k)} g(\mathbf{x}^{(k)}), \tag{3.32}$$

which is the MMSE estimate. Alg. 1 describes the procedure of SIS particle filter.

The SIS particle filter method suffers from the degeneracy problem [24]; eventually only one sample will have a dominant weight after a number of iterations. Various types of alternative particle filters [24, 28] were proposed to overcome the degeneracy problems. The sequential (or sampling) importance resampling (SIR) particle filter [27] is the most commonly used alternative in object tracking applications. The SIR particle filter is an extended version of the SIS particle filter. The main difference

Algorithm 1 SIS Particle Filter

Input: Random measure of previous posterior PDF, $\{\mathbf{x}_{t-1}^{(k)}, w_{t-1}^{(k)}\}_{k=1}^{K}$.
Input: Observation at t, $\mathbf{y}_t$.
Output: Random measure of current posterior PDF, $\{\mathbf{x}_{t}^{(k)}, w_{t}^{(k)}\}_{k=1}^{K}$.
Output: MMSE estimate of current state, $E(\mathbf{X}_t|\mathbf{y}_{0:t})$.

1: Do sampling: $\mathbf{x}_t^{(k)} \backsim q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)}, \mathbf{y}_t)$.
   // Prediction
2: Compute $p(\mathbf{x}_t^{(k)}|\mathbf{y}_{0:t-1})$ as in (3.27).
   // Update
3: Compute likelihood weights.
4: Obtain random measure of posterior probability.
   // Estimation
5: Compute MMSE estimate.

between SIR and SIS particle filters is that in the SIR particle filtering, by sampling the estimated posterior PDF, a new equally weighted posterior random measure is generated, which is known as the resampling process:

$$\left\{\tilde{\mathbf{x}}_t^{(k)}, \frac{1}{K}\right\}_{k=1}^{K} \Leftarrow \{\mathbf{x}_t^{(k)}, w_t^{(k)}\}_{k=1}^{K}. \tag{3.33}$$

This new random measure of the posterior PDF will not propagate the current weights to the next prediction step. Hence, it can be robust to the degeneracy problem; the weights of support points of posterior PDF will not be computed in a cumulative manner.

$$w_t^{(k)} \propto \frac{p(\mathbf{y}_t^{(k)}|\mathbf{x}_t^{(k)})p(\mathbf{x}_t^{(k)}|\tilde{\mathbf{x}}_{t-1}^{(k)})}{q(\mathbf{x}_t^{(k)}|\tilde{\mathbf{x}}_{t-1}^{(k)}, \mathbf{y}_t)}. \tag{3.34}$$

Also, in particle sampling, the SIR particle filter normally utilizes the transition density $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ as a proposal distribution:

$$\mathbf{x}_t^{(k)} = f_t(\tilde{\mathbf{x}}_{t-1}^{(k)}) + \mathbf{u}_t \Leftarrow p(\mathbf{x}_t|\mathbf{x}_{t-1}). \tag{3.35}$$

Algorithm 2 SIR Particle Filter

Input: Random measure of previous posterior PDF, $\{\tilde{\mathbf{x}}_{t-1}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$.
Input: Observation at t, $\mathbf{y}_t$.
Output: Random measure of current posterior PDF, $\{\tilde{\mathbf{x}}_{t}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$.
Output: MMSE estimate of current state, $E(\mathbf{X}_t|\mathbf{y}_{0:t})$.

1: Do sampling as in (3.35)
2: Compute likelihood weights as in (3.36).
3: Obtain random measure of posterior probability.
   // Estimation
4: Compute MMSE estimate.
5: Do resampling: Obtain new random measure, $\left\{\tilde{\mathbf{x}}_{t}^{(k)}, \frac{1}{K}\right\}_{k=1}^{K}$.

Then, the likelihood weights are simply formed as

$$w_t^{(k)} \propto p(\mathbf{y}_t^{(k)}|\mathbf{x}_t^{(k)}). \tag{3.36}$$

The procedure of the SIR particle filter is shown in Alg. 2.

3.2 Multiple Camera Particle Filter

Given L cameras, we denote the observation random vector from the i^{th} camera at time t as $\mathbf{Y}_{i,t}$ and its realization as $\mathbf{y}_{i,t}$. The random process of an object state at time t is denoted as $\mathbf{X}_t$ and its realization as $\mathbf{x}_t$. Fig. 3.1 illustrates the overview of a multiple camera based object tracking system. In the figure, $\mathbf{y}_{i,t}$ represents an extracted object feature, e.g., a color histogram, from an image captured at the i^{th} camera at time t. The dynamic model of the object is defined by a state process model:

$$\mathbf{X}_{t+1} = f_t(\mathbf{X}_t, \mathbf{U}_t). \tag{3.37}$$

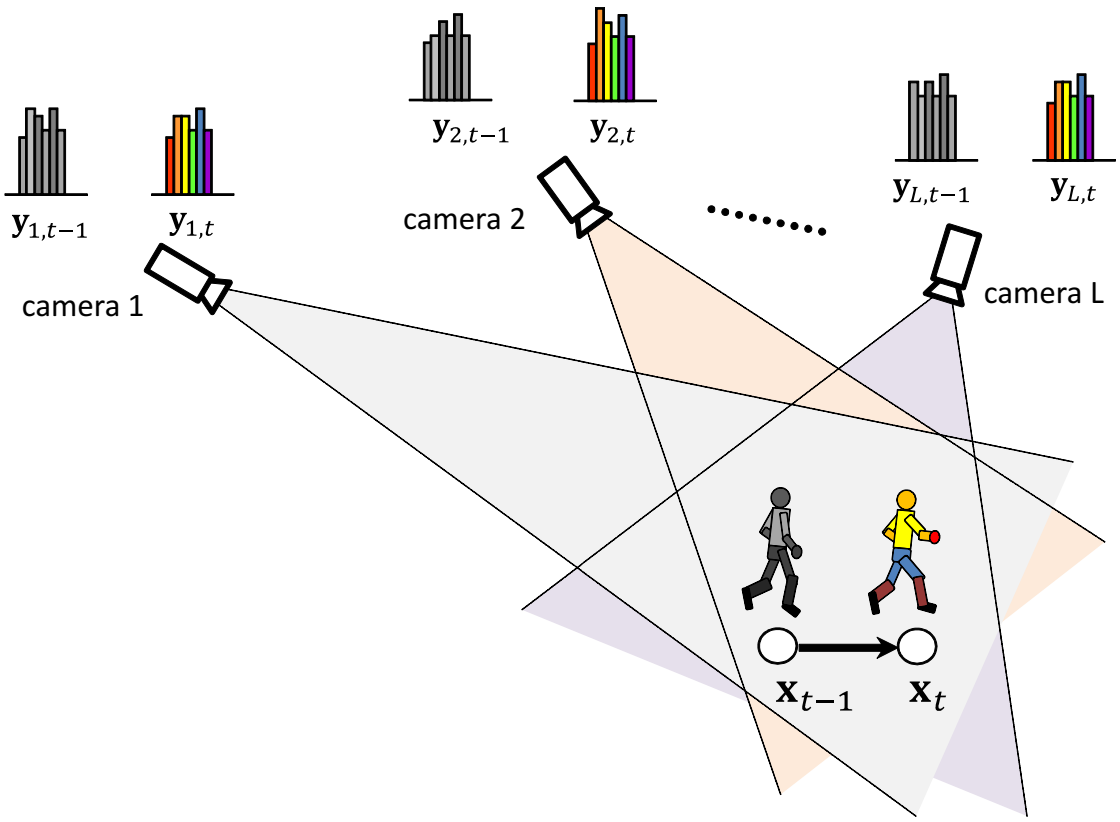


Figure 3.1.: Overview of a multiple camera based object tracking system.

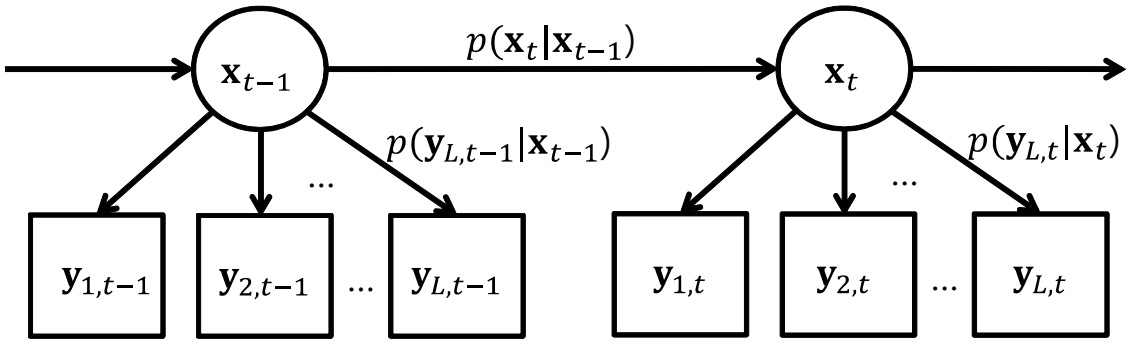


Figure 3.2.: Dynamic Markov model for multiple observation based tracking.

The observation model, assuming the observations are independent given the state, is defined as

$$\begin{aligned} \mathbf{Y}_{1,t} &= h_{1,t}(\mathbf{X}_t, \mathbf{V}_{1,t}) \\ &\vdots \\ \mathbf{Y}_{L,t} &= h_{L,t}(\mathbf{X}_t, \mathbf{V}_{L,t}), \end{aligned} \tag{3.38}$$

where $\mathbf{U}_t$ and $\mathbf{V}_{i,t}$ are independent and identically distributed (IID) white noise processes, and f_t and $h_{i,t}$ are assumed to be known functions. Also, each observation model can be characterized in probability by the associated likelihood

$$\mathbf{Y}_{i,t} = h_{i,t}(\mathbf{X}_t, \mathbf{V}_{i,t}) \Leftrightarrow p(\mathbf{y}_{i,t}|\mathbf{x}_t). \tag{3.39}$$

Then, the likelihood probability density function (PDF) given all the observations has the form

$$p(\mathbf{y}_t|\mathbf{x}_t) = \prod_{i=1}^{L} p(\mathbf{y}_{i,t}|\mathbf{x}_t), \tag{3.40}$$

where $\mathbf{y}_t = ((\mathbf{y}_{1,t})^T, \cdots, (\mathbf{y}_{L,t})^T)^T$.

Fig. 3.2 shows the graphical model of the dependencies between the object state and the L camera observations. The graphical model illustrates the evolution of the

Algorithm 3 Multiple observation based particle filter.

Input: Random measure of previous posterior PDF, $\{\tilde{\mathbf{x}}_{t-1}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$.
Input: Observations at t, $\mathbf{y}_{1,t}, \cdots, \mathbf{y}_{L,t}$.
Output: Random measure of current posterior PDF, $\{\tilde{\mathbf{x}}_{t}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$.
Output: MMSE estimate of current state, $E(\mathbf{X}_t|\mathbf{y}_{0:t})$.

1: Do sampling: $\mathbf{x}_t^{(k)} = f_t(\tilde{\mathbf{x}}_{t-1}^{(k)}) + \mathbf{u}_t$.
2: Compute (local) likelihood weights: $w_{i,t}^{(k)} = p(\mathbf{y}_{i,t}|\mathbf{x}_t^{(k)})$.
3: Compute joint (global) likelihood weight: $w_t^{(k)} = \prod_{i=1}^{L} w_{i,t}^{(k)}$.
4: Compute MMSE estimate: $E(\mathbf{X}_t|\mathbf{y}_{0:t}) = \sum_{k=1}^{K} w^{(k)}\mathbf{x}^{(k)}$.
5: Do resampling: $\left\{\tilde{\mathbf{x}}_t^{(k)}, \frac{1}{K}\right\}_{m=1}^{K} \Leftarrow \{\mathbf{x}_t^{(k)}, w_t^{(k)}\}_{k=1}^{K}$.

system over time as a hidden Markov dynamic model. The directed link from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ represents the state transition process in Eq. (3.37) with its associated probability, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The directed link from $\mathbf{x}_t$ to $\mathbf{y}_{i,t}$ represents the local observation process at the i^{th} camera in Eq. (3.39).

In this work, the SIR particle filter is utilized because it is simple, avoids the particle degeneration problem, and is commonly used in many visual object tracking applications. At each time instant, a tracking result is obtained by the minimum mean squared error (MMSE) estimate of the target state. Alg. 3 describes the SIR particle filter utilizing multiple independent observations. In Alg. 3, $\mathbf{x}_t^{(k)}$ and $\tilde{\mathbf{x}}_t^{(k)}$ are the k^{th} particle and resampled particle samples at time t, respectively. A random measure, $\{\mathbf{x}^{(k)}, w^{(k)}\}_{k=1}^{K}$, represents a probability in which

$$p(\mathbf{x}) \approx \sum_{k=1}^{K} w^{(k)}\delta(\mathbf{x} - \mathbf{x}^{(k)}). \tag{3.41}$$

As previously mentioned, after resampling, all the particles $\tilde{\mathbf{x}}_t^{(k)}$ have the same probability $\frac{1}{K}$, but there is a larger concentration of particles in regions of higher probability.

3.3 Distributed Particle Filter in Wireless Camera Networks

Wireless camera networks consist of multiple smart cameras, each equipped with an imaging sensor, a low-power processor, and a wireless transceiver. In such networks, low processing power and unreliable communication networks are the main challenges in developing sophisticated multiple view based tracking algorithms. Li [29] presented a camera activation scheme for object tracking, which is purposed to save camera sensor energy in WCNs. Assuming a large number of camera sensors is deployed with redundancy in camera views, the activation scheme makes visually correlated cameras simultaneously activate for target tracking. This approach saves the camera sensor energy consumed for sensing, data processing, and communication by making uncorrelated camera nodes inactive. Shin [30] proposed a predictive duty cycling method, which controls radio duty cycle and camera sensing rate. In his method, camera sensor nodes become activated when the predicted target location shows high probability in their sensing ranges. This predictive duty cycling approach reduces camera sensor energy consumption for sensing and data communication in object tracking. These approaches considered the mechanisms of camera sensor activation for energy saving in WCNs.

Fleck *et al.* [31] presented a distributed particle filter for multi-camera object tracking. To reduce the computational burden and communication bandwidth, each camera unit performs individual particle filtering, and transmits particle samples instead of raw image data. However, the particle samples, although the data size of the samples is much less than the raw image data, requires large communication bandwidth for the complex object probability and high dimensional object state. This approach may not be suitable for a mote based WCN, which has tight communication bandwidth. Qu *et al.* [32] proposed a distributed multiple target tracking method using multiple camera collaboration with sequential Monte Carlo implementation. They

illustrate the proposed multiple camera tracking method in a two-camera scenario using the epipolar geometry of the two cameras. In their method each camera unit operates its own tracker independently. Camera collaboration is only requested when occlusion occurs or targets are close to each other in a camera unit. The camera collaboration is done by computing the corresponding appearance measure in the other camera unit given a state (object position) in one camera. They proposed a heuristic approximation of the likelihood probability (appearance measure) using the distance between the state in one camera and the epipolar line of the estimated object position by the other cameras which do not exchange all particles in the networks. However, to perform the camera collaboration properly, the other cameras should track the target reliably, and the tracking result should be available. Furthermore, this approach is not able to increase each camera unit tracking accuracy, since each camera unit runs a single camera based tracking without camera collaboration in non-occlusion cases.

In addition to the referred distributed particle filters for camera networks, there are several distributed particle filtering approaches [9,33–36] for wireless sensor networks. In the following sub-section, we review distributed particle filter implementations, considering a WCN as a special wireless sensor network. Then, we describe cluster-based distributed particle filter implementations, which are designed for WCNs.

3.3.1 Distributed Particle Filter in Wireless Sensor Networks

In a centralized wired sensor network consisting of one computational unit and multiple sensing units, Alg. 3 whose input arguments are all sensor observations may be applicable; each sensor sends its observation, e.g., mostly sonar or image data to the central unit, and then the central unit does object tracking. However, this centralized approach requires a certain amount of bandwidth to transmit observations to the central unit and the real time computation power of the central unit to run the

tracking algorithm with all sensed information. In the implementation of distributed particle filters communication related costs are a major concern, since most energy is consumed in communications and data communication bandwidth is also limited. Transmitting raw sensing data or even particle information (set of particles and associated weights) in such networks is undesirable. The majority of the distributed particle filter implementations in sensor networks hence focuses on minimizing the communication costs. In this section, we will review currently proposed distributed implementations in terms of communication and computation aspects.

The distributed particle filter implementations can be categorized into two types - observation related information exchanging type and probability related information exchanging type.

- Distributed implementation type 1: Observations, $\mathbf{y}_{i,t}$, are exchanged.
- Distributed implementation type 2: Probability representations, $p(\mathbf{y}_t|\mathbf{x}_{i,t})$ or $p(\mathbf{x}_{i,t}|\mathbf{y}_{0:t})$, are exchanged.

Let us say a sensor node consists of both computational and sensing units, and the sensor runs a tracking algorithm using its own observations and each sensor node will keep track of an object using all other sensor node information. The first type implementation basically utilizes Alg. 3. In this approach, the local particle filter sends its observation, $\mathbf{y}_{i,t}$, through the network and receives all other sensors' observations to run the generic multiple observation based particle filter. The papers [9,34] tackled the first type of distributed particle filter. Local sensors communicate with each other using compressed observation data to reduce communication load instead of transmitting raw observation through the network. It can reduce bandwidth but it increases the amount of computations; it is needed to run data compression and decompression processes which may require significant computational power. The procedure of this type of implementation is shown in Alg. 4.

Algorithm 4 Compression based distributed particle filter at i^{th} sensor.
Given: A network with L sensors.

Do compression: $\tilde{\mathbf{y}}_{i,t} \Leftarrow \mathbf{y}_{i,t}$.
Transmit: Observation at t to other sensors, $\tilde{\mathbf{y}}_{i,t}$.

Receive: Observations at t from other sensors, $\tilde{\mathbf{y}}_{j,t}$, $j = 1, \cdots, L$, $j \neq i$.

Do decompression:$\mathbf{y}_{j,t} \Leftarrow \tilde{\mathbf{y}}_{j,t}, j = 1, \cdots, L, j \neq i$.
Perform Algorithm 3.

In the second type of implementation, each local tracker exchanges its probability representation to perform global particle filtering. As shown in Alg. 3, the collaborative work is done by computing the joint likelihood. Hence, it is possible for a sensor unit to build the global probability by obtaining local likelihoods from all other sensors. For distributed particle filtering using multiple cameras, each camera computes its object probability locally, and the global object probability is obtained by fusing the local object probabilities of the camera nodes. In this case, however, the discrete nature of the method requires particle sample synchronization in computing the joint (global) probability [9, 33, 35, 36]. Note that we use the term *synchronization* to describe the fact that all local particle sets have the same support points as defined in [9]. Fig. 3.3 shows the synchronization problem when two different camera nodes must build a joint probability. In the figure, each vertical bar represents a weighted particle sample. As shown on the left side of the figure, local probabilities corresponding to asynchronous particles from different cameras, which have different support points in their probability representations, lead to inaccurate joint probability estimates. Furthermore, as the erroneous probability propagates frame by frame, it deteriorates the recursive distributed particle filtering and causes severe tracking performance degradation. In distributed particle filter implementations, one of the main challenges is to make all the camera particles synchronized [9, 33, 37, 38].

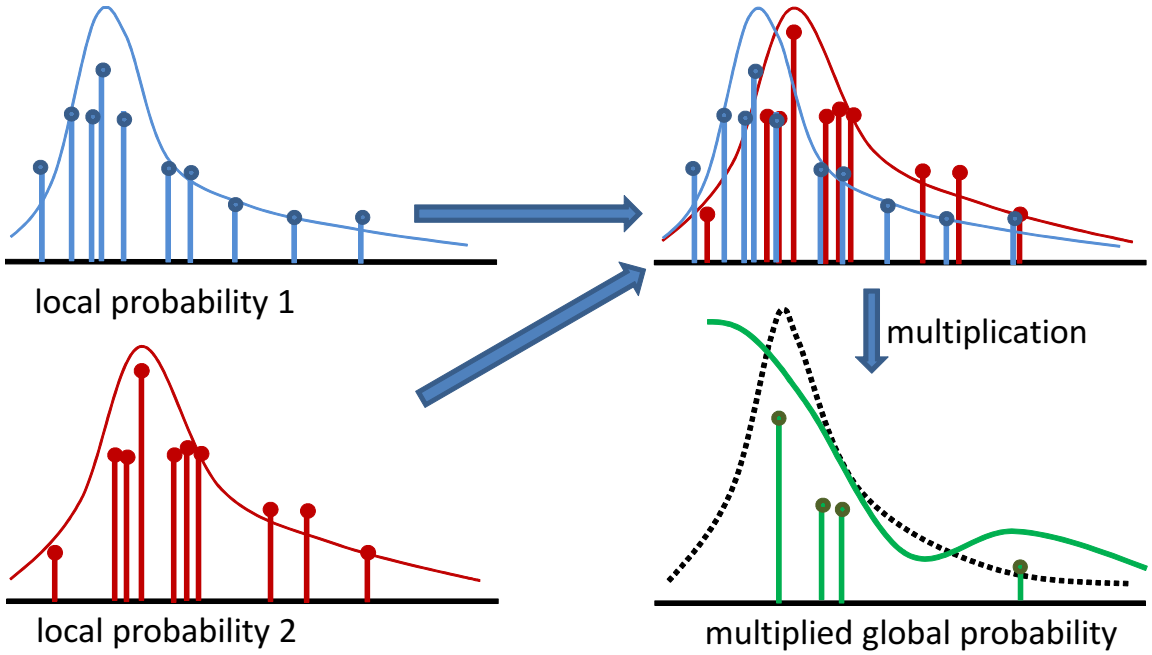


Figure 3.3.: Particle synchronization problem to compute a joint probability. The top-right plot shows the overlapped local probabilities and the bottom-right plot illustrates the multiplied probability of the two local probabilities. In the global probability plot, the black dotted line indicates the desired probability density and the solid green line shows the global probability computed by the multiplication of two asynchronous discrete local probabilities.

We consider two approaches for handling the particle synchronization issue. The first approach, which is an academic exercise to show the effects of perfect particle synchronization, synchronizes all local particles by forcing the local random number generators to use the same random seed. The second and more practical approach is to convert discrete probabilities to continuous forms that do not require synchronization. We will refer to the first approach as *synchronized particle filtering*, and to the second as *probability conversion based particle filtering*. Before describing these two approaches, we first introduce a general multiple camera particle filtering framework, which lays the groundwork for our distributed particle filter implementations.

3.3.1.1 Synchronized Particle Filter

As shown in Alg. 3, the collaborative work of the multiple observation based particle filter is carried out by computing the joint likelihood weights $w_t^{(k)}$. Let us say the k^{th} weight of the i^{th} camera, $w_{i,t}^{(k)}$, represents the likelihood at the k^{th} support point, $\mathbf{x}_{i,t}^{(k)}$. The k^{th} support points of all cameras should be the same to obtain the joint probability by the multiplication of all camera weights. If any of the local likelihood weights have different support points, the multiplication of local weights does not guarantee the correct joint probability.

In the synchronized distributed particle filter [39], all local particle filters are synchronized to have the same support points. To make all camera particles synchronized, all cameras are restricted to use a common random seed to initialize the random number generators responsible for the sampling and resampling processes, which is equivalent to operating all local cameras with a single set of particles. When all local particle filters are synchronized with the same random seed, multiple observation based particle filtering can be accomplished in a distributed way by communicating

Algorithm 5 Synchronized particle filter at node i.

Given: A network with L sensors.

1: Do sampling with a common random seed
2: Compute local likelihood weights: $w_{i,t}^{(k)} = p(\mathbf{y}_{i,t}|\mathbf{x}_t^{(k)})$.
3: Transmit local random measure (weights): $\{w_{i,t}^{(k)}\}_{k=1}^{K}$.
4: Receive other sensors' random measures (weights): $\{w_{j,t}^{(m)}\}_{k=1}^{K}$, $j \neq i$.
5: Compute joint global global weights: $w_t^{(k)} = \prod_{j=1}^{L} w_{j,t}^{(k)}$.
6: Compute posterior PDF
7: Compute MMSE estimate
8: Do resampling with the same random seed

only local weights. Alg. 5 shows the general procedure of the synchronized particle filter at a given node.

3.3.1.2 Probability Conversion Based Particle Filter

When the synchronized particle filter is utilized in a camera network, the amount of data transmitted by a camera node is solely dependent upon the number of particles. For networks that have a tight communication bandwidth, it is indispensable that all nodes operate with a small number of particles to prevent tracking performance degradation. Also, when an event breaks the particle synchronization, e.g. a failure to share the seed due to communication loss, the multiple node collaboration cannot take place properly, which deteriorates the tracking performance. Hence, in many cases, it is preferable to encode the probability representation into a continuous form that is independent of the number of particles and does not require particle synchronization. Gaussian mixture model (GMM) [10, 11, 40–45] and Parzen [33, 35–37] based probability conversion methods were proposed for converting particle probabilities into continuous forms. In this section, we show the derivation of a distributed particle filtering framework using the aforementioned probability conversion approaches, which will be utilized for the proposed tracking system.

For convenience of presentation, we first define some additional notation before describing the probability conversion based particle filters. Let us say $\Gamma(\mathbf{x}_t)$ is the prior PDF, $p(\mathbf{x}_t|\mathbf{y}_{0:t-1})$, in the prediction step

$$\Gamma(\mathbf{x}_t) \triangleq p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{0:t-1}). \tag{3.42}$$

Additionally, let us denote a local posterior PDF constructed from a local likelihood PDF at the i^{th} node (camera) as $\Delta_i(\mathbf{x}_t)$:

$$\Delta_i(\mathbf{x}_t) \triangleq \frac{1}{Z}p(\mathbf{y}_{i,t}|\mathbf{x}_t)\Gamma(\mathbf{x}_t), \tag{3.43}$$

where Z is the normalization constant. Then, the global posterior probability can be computed at camera i in an L camera network as

$$p(\mathbf{x}_t|\mathbf{y}_{0:t}) = \Delta_i(\mathbf{x}_t)\prod_{j\neq i}^{L} p(\mathbf{y}_{j,t}|\mathbf{x}_t). \tag{3.44}$$

For all $j \neq i$, the local likelihood $p(\mathbf{y}_{j,t}|\mathbf{x}_t)$ can be computed at node i if the posterior $\Delta_j(\mathbf{x}_t)$ and the prior PDF $\Gamma(\mathbf{x}_t)$ are known by node i, since

$$p(\mathbf{y}_{j,t}|\mathbf{x}_t) \propto \frac{\Delta_j(\mathbf{x}_t)}{\Gamma(\mathbf{x}_t)}. \tag{3.45}$$

By exchanging the local posterior PDFs, $\Delta_j(\mathbf{x}_t)$, each node can compute the likelihoods of the other cameras using Eq. (3.45), and then build the global posterior PDF using Eq. (3.44). This procedure effectively allows for object tracking to be performed in a distributed manner.

Assuming we have a global random measure $\{\tilde{\mathbf{x}}_{i,t-1}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$ of the previous global posterior PDF at the i^{th} camera, we can compute the prior PDF at the i^{th} camera by the following sampling procedure:

$$\mathbf{x}_{i,t}^{(k)} = f_t(\tilde{\mathbf{x}}_{i,t-1}^{(k)}) + \mathbf{u}_{i,t},\ k = 1, \cdots, K, \tag{3.46}$$

$$\Gamma_i(\mathbf{x}_t) \approx \frac{1}{K}\sum_{k=1}^{K}\delta(\mathbf{x}_t - \mathbf{x}_{i,t}^{(k)}), \tag{3.47}$$

where $\Gamma_i(\mathbf{x}_t)$ is the prior PDF at the i^{th} camera. Note that $\Gamma_i(\mathbf{x}_t)$ is equivalent to $\Gamma(\mathbf{x}_t)$ for all i, since all $\Gamma_i(\mathbf{x}_t)$ represent the same prior PDF with different support points (recall that in the probability conversion approaches we no longer require synchronized particles). The local likelihood weights at camera i are formed as

$$w_{i,t}^{(k)} = \frac{1}{Z}p(\mathbf{y}_{i,t}|\mathbf{x}_{i,t}^{(k)}), \tag{3.48}$$

where Z is the normalization constant. Then, the local posterior PDF, which is characterized by the random measure $\{\mathbf{x}_{i,t}^{(k)}, w_{i,t}^{(k)}\}_{k=1}^{K}$, is expressed as

$$\Delta_i(\mathbf{x}_t) \approx \frac{1}{K}\sum_{k=1}^{k} w_{i,t}^{(k)}\delta(\mathbf{x}_t - \mathbf{x}_{i,t}^{(k)}). \tag{3.49}$$

After computing the local posterior PDF, it is necessary to convert this discrete local posterior PDF form into a continuous representation. In the GMM representation, the local posterior PDF is converted to an equally weighted random measure by the resampling process; then GMM parameters (mixture component weights, means, and variances) are computed to form the GMM representation in which

$$\Delta_i(\mathbf{x}_t) \approx \frac{1}{Z_c}\sum_{n=1}^{N_{GMM}} c_{i,t}^{(n)}\mathcal{N}(\mathbf{x}_t - \boldsymbol{m}_{i,t}^{(n)}, \boldsymbol{\Sigma}_{i,t}^{(n)}), \tag{3.50}$$

where Z_c is the normalization constant, N_{GMM} is the number of GMM mixture components, $c_{i,t}^{(n)}$ is the n^{th} mixture component weight, and $\mathcal{N}(m, \Sigma)$ is a Gaussian PDF with mean, m, and variance, Σ. The procedure of GMM density estimation is explained in Appendix A.

An alternative approach to represent the particles using a continuous distribution is through a Parzen window representation. For Parzen form conversion, we draw N_{PZ} samples $\{\bar{\mathbf{x}}^{(n)}\}_{n=1}^{N_{PZ}}$ from the local posterior PDF. Then, we have the random measure $\{\bar{\mathbf{x}}_{i,t}^{(n)}, \frac{1}{N_{PZ}}\}_{n=1}^{N_{PZ}}$ that is directly transformed to a continuous PDF representation. Note that the number of samples N_{PZ} does not need to be the same as the number of particles K to represent the local posterior PDF in continuous form. In fact, the continuous PDF representation of particle samples using Parzen windows requires significantly fewer samples than the corresponding discrete approximation, i.e., $N_{PZ} \ll K$. The continuous form of the local posterior PDF from the samples is expressed as

$$\Delta_i(\mathbf{x}_t) \approx \frac{1}{Z_k} \sum_{n=1}^{N_{PZ}} P_i(\mathbf{x}_t - \bar{\mathbf{x}}_{i,t}^{(n)}), \tag{3.51}$$

where Z_k is the normalization constant and $P_i(\cdot)$ is a Parzen's kernel. The procedure of Parzen density estimation is explained in Appendix B.

After the conversion to continuous form of the local posterior PDF, each camera exchanges its converted probability in the form of the Parzen's samples $\{\bar{\mathbf{x}}_{i,t}^{(n)}\}_{n=1}^{N_{PZ}}$ or the GMM parameters $\{c_{i,t}^{(n)}, \boldsymbol{m}_{i,t}^{(n)}, \boldsymbol{\Sigma}_{i,t}^{(n)}\}_{n=1}^{N_{GMM}}$. Then, the local likelihood weights of other nodes are computed at the support points of the i^{th} camera prior PDF as shown in Eq. (3.45), with $\Delta_j(\mathbf{x}_{i,t}^{(k)})$ given by Eq. (3.50) or (3.51), i.e.:

$$w_{j,t}^{(k)} \triangleq \frac{\Delta_j(\mathbf{x}_{i,t}^{(k)})}{\Gamma(\mathbf{x}_{i,t}^{(k)})}, \text{ for } j \in \{1, \cdots L\}. \tag{3.52}$$

Then, the global likelihood weight can be computed as

$$w_t^{(k)} = \frac{\prod_{j=1}^{L} w_{j,t}^{(k)}}{\sum_{k=1}^{K} \prod_{j=1}^{L} w_{j,t}^{(k)}}. \tag{3.53}$$

The new random measure, $\{\mathbf{x}_{i,t}^{(k)}, w_t^{(k)}\}_{k=1}^{K}$, characterizes the global posterior PDF. Finally, by applying resampling, each camera obtains the equally weighted random measure, $\{\tilde{\mathbf{x}}_{i,t}^{(k)}, \frac{1}{K}\}_{k=1}^{K}$ for the next estimation.

3.3.1.3 Computational Load in Each Camera Node

The Parzen method based particle filter requires additional sampling and reconstruction steps compared to the synchronized particle filter. The additional complexity of the Parzen method due to both sampling and reconstruction is $O(K_s)$, where K_s is the number of the particle samples. However, the GMM approach needs the mixture model training phase using the k-means clustering algorithm and EM (expectation maximization). Let N_{EM} denote the number of EM iterations, N_{GMM} the number of mixtures, and D_s the sample dimension, then $O(N_{EM}N_{GMM}D_s)$ is a rough estimation of the computational complexity of using EM to train a GMM [46] and the complexity of k-means is $O(N_{GMM}K_s)$. Generally, K_s is much larger than $N_{GMM}D_s$. Therefore, the GMM method using large sample data has roughly $O(N_{GMM}K_s)$ complexity. That is approximately N_{GMM} times to the Parzen method.

4. RESOURCE-AWARE DISTRIBUTED PARTICLE FILTER IN WIRELESS CAMERA NETWORKS

When developing a distributed tracking method, it is important to maximize the tracking system efficiency under given resource constraints such as communication bandwidth, processor's computational power, and energy consumption. Let us consider the constraints that affect the tracking algorithm design and application. While available communication bandwidth is dependent on network traffic, computational capacity for running a tracking algorithm is specified by hardware, which can be treated, to a great extent, as a static resource. Hence, the number of particle samples, which is the main factor influencing computational power consumption, can be set as a pre-assigned value according to the hardware specification. The computational resources required for carrying out a tracking algorithm typically do not change significantly over the course of tracking. Communication resources, on the other hand, are much more dynamic.

In this dissertation, we consider dynamically available communication resources and propose a resource-aware method, which reduces communication failures thereby improving the distributed tracking performance. Even though existing network protocols for wireless sensor networks, such as STCP [47], Fusion [48], CODA [49], and PCCP [50], control traffic congestion and reduce the chance of communication failures by adjusting the data transmission rate, these methods do not affect the number of data packets generated by a sensor node. Instead, they control the transmission rate of queued data. The proposed resource-aware method computes the number of communication packets that are available for data transmission at a given time ac-

cording to the network data traffic conditions and hence allows the particle filter to dynamically adjust the quality (resolution) of the tracking data to be communicated.

In this chapter, we first discuss design challenges imposed by WCNs in details in Section 4.1. Then, we introduce a novel distributed particle filter approach that is based on a resource-aware method for cluster-based WCNs. We utilize the dynamic clustering protocol in [7] as the collaborative processing framework for our implementation. The clustering protocol describes a mechanism to form clusters for camera collaborations. Let us say a camera cluster is formed with the purpose of tracking an object, so that a camera node in the cluster is elected as the cluster head, and the other camera nodes are assigned as cluster members. As described in [7, 8], we assume that cluster members are one hop neighbors of the cluster head but are not necessarily within single hop communication range of one another. In this environment, the cluster head estimates the global posterior probability of the target object by combining the local information transmitted from member cameras. Additionally, the cluster head broadcasts the joint posterior probability to its members, so that all the cluster members can perform the object tracking task with the same global information. An overview of the cluster-based tracking approach is shown in Fig. 4.1.

4.1 Challenges in Wireless Camera Networks

Design specifications for computer vision algorithms such as object tracking for WCNs are constrained by the characteristics of the camera node hardware, communication channel, network topology, network traffic, etc. In this section, we review the issues which need to be considered when developing vision algorithms for WCNs.

Let us first consider design challenges caused by hardware. Wireless smart cameras consisting of sensing, data processing, and communication units impose constraints on image quality, computational complexity of the vision algorithm, and data com-

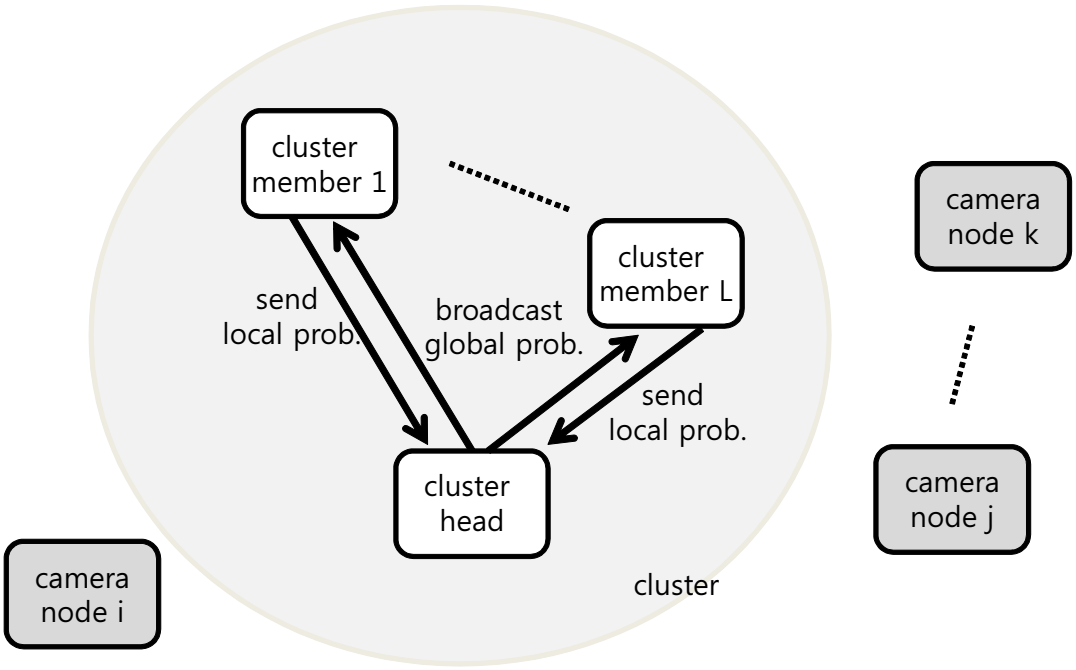


Figure 4.1.: Data communication in cluster-based distributed particle filter.

munication bandwidth. This is particularly true when severely resource-constrained mote-based embedded systems such as the Imote2 are utilized as wireless smart cameras. Computational limitations prevent smart camera systems from executing complicated vision algorithms, including most current state-of-the-art approaches. Low communication bandwidth and limited transmission energy available in a sensor node make it difficult to transmit raw image data and even sets of visual feature data. We can consider lightweight and distributed computer vision processing as one of the most promising approaches for reducing the computational burden in individual nodes. However, it is crucial to develop distributed methods that require only a small amount of data communication.

WCNs often suffer from unreliable communication, as wireless communication channels suffer from effects such as Rayleigh fading and inherently impose packet losses [51]. Unreliable communication causes data losses and degrades the overall quality of the received visual information, consequently leading to inaccurate vision processing at the camera nodes. Hence, in designing computer vision algorithms for WCNs, it is necessary that the distributed vision processing be robust to imperfectly communicated data.

In applying a vision task for WCNs, it is also important to consider mechanisms for collaborative processing. A cluster-based approach as suggested in [7,8] allows camera networks to work collaboratively. This collaborative processing causes additional data traffic on the clustered nodes when an event of interest is detected by a cluster. Intensive data traffic in a cluster tends to cause packet collisions and additional packet losses, and has a degrading effect on the performance of the vision tasks. Fig. 4.2 shows typical throughput and packet loss rate curves in wireless networks. As shown in Fig. 4.2 (a), there is a limit to the throughput that can be obtained by the network.

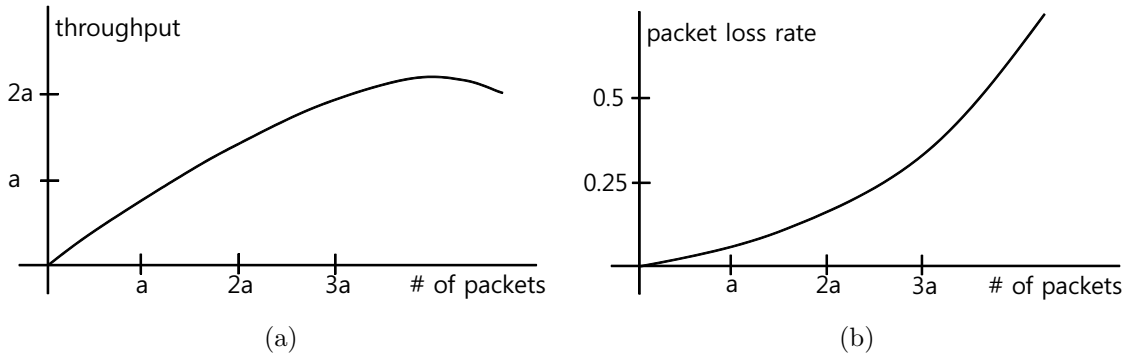


Figure 4.2.: Typical characteristics of wireless network traffic. (a) throughput vs. number of transmitted packets and (b) packet loss rate vs. transmitted packets.

This is essentially caused by the fact that, as shown in Fig. 4.2 (b), the packet loss rate increases sharply as more packets are transmitted in the network.

Data congestion and data loss are handled, in general, by reliable transport protocols, quality of service (QoS) mechanisms or by over-provisioning network capacity. Transport protocols such as TCP, however, are not suitable for delay sensitive applications, e.g., real time tracking applications, because such protocols as well as scheduling based congestion control protocols can cause excessive latency [52]. An approach of over-provisioning network bandwidth for the expected peak traffic load is not usually applicable to resource constrained wireless camera nodes either, since the nodes have tightly limited hardware capabilities. Network protocols for wireless sensor networks such as CSMA/CA (carrier sense multiple access with collision avoidance) [53], S-MAC [54], and T-MAC protocols [55], were proposed to reduce data contention. However, these contention-based schemes can not mitigate traffic congestion when the traffic exceeds a node buffer capacity. Many congestion control protocols also proposed to alleviate such traffic congestion for wireless sensor networks. Iyer *et al.* [47] presented a sensor transmission control protocol (STCP), which utilizes a retransmission scheme. It is not suitable for delay-sensitive applications, such as WCNs. Hull *et*

al. [48] proposed a congestion protocol, *Fusion*, which controls traffic congestion by limiting packet forwarding when congestion is detected and notified. Wan *et al.* [49] presented a energy efficient congestion control, CODA, which adjusts packet sending rate using packet drop or AIMD (Additive Increase Multiplicative Decrease) mechanism. Wang *et al.* [50] proposed a node priority-based congestion control protocol (PCCP), which adjusts the scheduling rate and source rate at each sensor depending on congestion degree and priority index of node. Fusion [48] and CODA [49] uses queue length at intermediate nodes to detect congestion. PCCP [50] utilizes packet inter-arrival time and packet service time to measure congestion degree.

In a WCN in which network capacity is limited, data loss during communication is inevitable. Vision applications for the wireless network need to be dynamically adaptable under varying traffic conditions, for example, by controlling the amount of the data transmitted without exceeding available bandwidth in the network.

4.2 Resource-Aware Packet Allotment

Under the assumption that tracking accuracy increases with the number of observations, tracking performance can be maximized when we utilize the information provided by all the members of a cluster. However, if we allow a large number of cluster members to transmit information at the maximum attainable data packet load, then the data traffic within the cluster increases, which may cause severe data loss, as mentioned in Section 4.1.

In this section, we present a resource-aware method that recognizes the data traffic conditions and computes the optimal amount of data that should be transmitted in a cluster. The optimal data packet load is obtained by maximizing the total amount of data transmitted in a cluster given the maximal allowed packet loss rate. This

procedure also confines the energy waste level of data transmission to the user-defined missing rate when transmitting local data with the optimal data packet load.

Let us say that R is the total data packet load, i.e. the total number of packets containing particle weights, Parzen samples or GMM parameters in a frame, and M is the packet loss rate. Then, the plot in Fig. 4.2 (b) can be expressed as a function $M(R)$ shown in Fig. 4.3. We obtain the optimal data packet load, R_{opt}, by solving the following problem:

$$\begin{aligned} R_{opt} &= max\{R(M)\} \\ s.t. &\quad M < M_{max}, \end{aligned} \tag{4.1}$$

where $R(M)$ is the inverse function of $M(R)$, and M_{max} is a user-defined packet loss rate. When $M(R)$ is a monotonically increasing function as shown in Fig. 4.3, $R(M)$ also becomes a monotonically increasing function. Then, the solution R_{opt} is in fact $R(M_{max})$. However, the rate function $R(M)$ is generally not available in practice but the packet loss rate can be measured at each time frame as

$$M(R_{TX}) = (R_{TX} - R_{RX})/R_{TX}, \tag{4.2}$$

where R_{TX} and R_{RX} are the number of transmitted and received packets, respectively. Thus, it is preferable to re-formulate the problem using the packet loss rate function.

Since $M(R_{opt}) = M_{max}$ as described in Fig. 4.3, we can re-formulate the optimization problem in Eq. (4.1) to find the optimal data packet load, R_{opt}, for M to be our target value, M_{max}. Using a convex cost function such as a squared difference function, the optimal data packet load problem can be expressed as

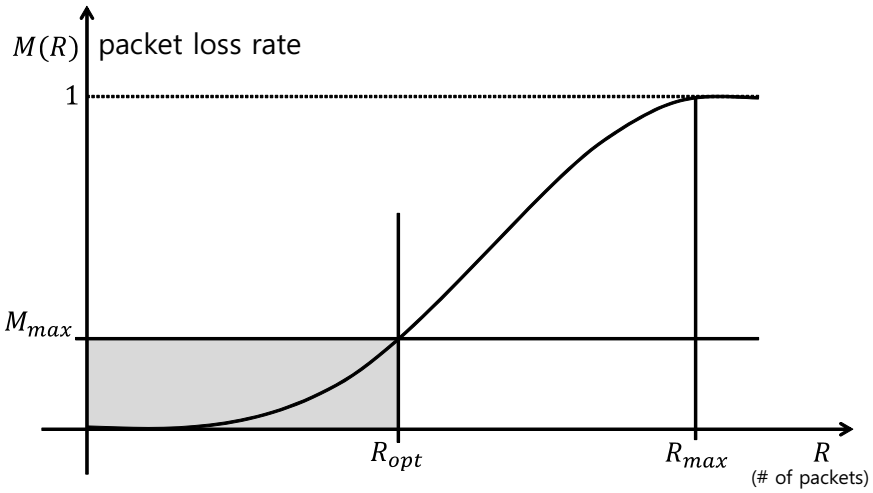


Figure 4.3.: Optimal data packet load point in the packet loss rate function.

$$\begin{aligned} R_{opt} &= \arg\min \left\{ (M(R) - M_{max})^2 \right\} \\ s.t. &\quad 0 < R < R_{max}, \end{aligned} \tag{4.3}$$

where R_{max} is the data packet load for which all the data is missed (i.e., M_{max} becomes 1). We assume that as R increases, the packet loss rate monotonically increases as in [56]:

$$M(R_{opt} - \delta) < M(R_{opt}) < M(R_{opt} + \delta), \tag{4.4}$$

where δ is a small number. Then, the cost function $(M(R) - M_{max})^2$ becomes a unimodal function, which has a single optimum. Hence, the minimizer R_{opt} can be obtained by applying a gradient descent method [57] and the iterative solution is

$$R_{opt}^{(t+1)} = R_{opt}^{(t)} - \mu_t \nabla_t, \tag{4.5}$$

Algorithm 6 Packet allotment procedure.

Given: the target packet loss rate, M_{max}, the minimum data packet load, R_{min}, and the data packet load at t, $R_{opt}^{(t)}$.

1: Count received data packets: $R_{RX}^{(t)}$.
2: Compute the packet loss rate:
$M(R_{opt}^{(t)}) = (R_{opt}^{(t)} - R_{RX}^{(t)})/R_{opt}^{(t)}$.
3: Update data packet load:
$R_{opt}^{(t+1)} = R_{opt}^{(t)} - \beta(M(R_{opt}^{(t)}) - M_{max})$.
4: Compute individual node data packet load:
$R_l^{(t+1)} = max\left\{\left\lfloor R_{opt}^{(t+1)}/L\right\rfloor, R_{min}\right\}$.

where $\nabla_t = M'(R_{opt}^{(t)})(M(R_{opt}^{(t)}) - M_{max})$, μ_t is the step size of the iteration, and $\mu_t > 0$. Since we cannot access the derivative of the packet loss rate function, $M(\cdot)$, we assign μ_t as $\beta_t/M'(R_{opt}^{(t)})$, where $\beta_t > 0$. Note that $M'(R_{opt}^{(t)})$ is positive since the packet loss rate monotonically increases in the operational range such that $0 < R < R_{max}$. Then the iterative equation becomes

$$R_{opt}^{(t+1)} = R_{opt}^{(t)} - \beta_t(M(R_{opt}^{(t)}) - M_{max}). \tag{4.6}$$

When we assign β_t as a constant, we have the following condition for convergence [57]

$$\beta < M'(R). \tag{4.7}$$

This condition indicates that β should be a small number if the slope of $M(R_{max})$ is small, and that there is a limit in the rate of change of $M(R)$ that the system is capable of handling.

Let us say the solution at the current frame, R_{opt}, is the available communication resource in a cluster. It is necessary to assign a proper data packet load for each node, R_l, where $\sum_{l=1}^{L} R_l = R_{opt}$. One may utilize a rate-distortion optimization approach [58], which assigns each node communication rate according to the difference between the transmitted and received probabilities of each node, caused by packet

loss. However, in hardware and bandwidth constrained smart camera applications, it may not be feasible to run the rate-distortion optimization process, since it requires both monitoring the data packet loads of all nodes and measuring their probability differences. Assuming the communication channels and packet loss rates of all camera nodes are approximately the same, we use a fair packet allotment approach, which is

$$R_l = max\left\{\left\lfloor\frac{R_{opt}}{L}\right\rfloor, R_{min}\right\}, \tag{4.8}$$

where R_{min} is a minimum packet load for data transmission, which prevents the rate from becoming zero. Note that the floor operator, $\lfloor\cdot\rfloor$, is used to ensure that the resulting packet load is an integer value. Alg. 6 describes the procedure of packet loss measurement and packet allotment.

4.3 Resource-Aware Distributed Particle Filter

In this section, we describe the proposed cluster-based distributed particle filter implementations. Alg. 7 and Alg. 8 illustrate the particle filtering procedures for the cluster members and the cluster head, respectively. For the sake of convenience, we divide the filtering procedure of a node into three sub phases: *pre-processing*, *data communication*, and *post-processing*.

In pre-processing and post processing, the nodes perform common particle filtering operations consisting of sampling, computing object likelihoods, resampling, and estimating the target position. At the data communication phase, the cluster head and cluster members carry out different operations. First, cluster members prepare for data communication by computing the amount of data load according to Eq. (4.8) and, when applicable, converting their probabilities into continuous forms (e.g., Parzen samples or GMM parameters) according to the data packet load; the num-

Algorithm 7 Cluster-based distributed particle filter at the i^{th} cluster member.

Given: the previous posterior PDF random measure, $\left\{\tilde{\mathbf{x}}_{i,t-1}^{(k)}, \frac{1}{K}\right\}_{k=1}^{K}$ and data packet load, $R_l^{(t-1)}$.

Pre-Processing

1: Do sampling: $\mathbf{x}_{i,t}^{(k)} = f_t(\tilde{\mathbf{x}}_{i,t-1}^{(k)}) + \mathbf{u}_{i,t}$.
2: Compute local weights: $w_{i,t}^{(k)}$.

Data Communication

1: Convert local probability into continuous form according to the data packet load $R_l^{(t-1)}$.
2: Transmit local probability to cluster head.
3: Receive global probability and the next data packet load $R_l^{(t)}$ from cluster head.
4: Reconstruct global posterior PDF (update weights): $w_t^{(k)}$.

Post-Processing

1: Do resampling: $\left\{\tilde{\mathbf{x}}_{i,t}^{(k)}, \frac{1}{K}\right\}_{m=1}^{K} \Leftarrow \{\mathbf{x}_{i,t}^{(k)}, w_t^{(k)}\}_{k=1}^{K}$.

Algorithm 8 Cluster-based distributed particle filter at the cluster head.

Given: the previous posterior PDF random measure, $\left\{\tilde{\mathbf{x}}_{i,t-1}^{(k)}, \frac{1}{K}\right\}_{k=1}^{K}$

Pre-Processing

1: Do sampling: $\mathbf{x}_{i,t}^{(k)} = f_t(\tilde{\mathbf{x}}_{i,t-1}^{(k)}) + \mathbf{u}_{i,t}$.
2: Compute local weights: $w_{i,t}^{(k)}$.

Data Communication

1: Receive local probabilities from cluster members.
2: Reconstruct local likelihood weights: $w_{j,t}^{(k)}$, $j \neq i$.
3: Compute joint global weights: $w_t^{(k)} = w_{i,t}^{(k)} \prod_{j \neq i}^{L} w_{j,t}^{(k)}$
4: Compute data packet load for next frame according to Alg. 6.
5: Transmit global probability and the data packet load to cluster members.

Post-Processing

1: Do resampling: $\left\{\tilde{\mathbf{x}}_{i,t}^{(k)}, \frac{1}{K}\right\}_{k=1}^{K} \Leftarrow \{\mathbf{x}_{i,t}^{(k)}, w_t^{(k)}\}_{k=1}^{K}$.
2: Compute MMSE estimate: $E(\mathbf{X}_t|\mathbf{y}_{0:t})$.

ber of Parzen samples or GMM parameters to be transmitted is proportional to the packet rate. Then the cluster head receives the local data from the cluster members and builds a joint probability. The cluster head also updates the total packet rate of the cluster by inspecting the number of missed packets at the current frame, as

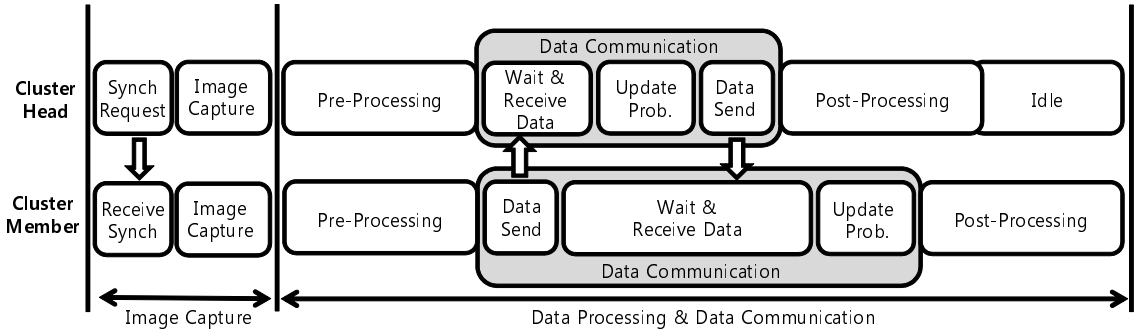


Figure 4.4.: Distributed particle filter timing diagram.

described in Alg. 6. Then, the cluster head broadcasts the global probability to the cluster members, which execute their particle filters with the received global information. The timing diagram of the cluster-based distributed particle filter, as shown in Fig. 4.4, describes one cycle of processing, including image capturing and data processing.

5. EXPERIMENTS

In this chapter, we present experiments that demonstrate the feasibility of the proposed method. We first describe the evaluation metrics we utilize for analyzing the proposed approaches. Then, the detailed experimental setting, including parameters of the particle filters, communication payloads, and packet loss models are introduced so that we can finally show the experimental results.

5.1 Evaluation Metrics

The proposed distributed tracking systems have two main functionalities: One is object tracking based on particle filtering, and the other is packet allotment, which is independent of the tracking mechanism. To evaluate the performance of the object tracking methods, two metrics are utilized, *tracking error* and *score*. In order to show the performance of the resource-aware packet allotment mechanism, the *delivery energy efficiency* metric is introduced. These metrics are described in detail in the following sections.

5.1.1 Tracking Evaluation Metrics

To evaluate the performance of a visual tracking system, a human visually verifies the tracking results by manually creating bounding circles or boxes surrounding the objects being tracked at each image frame. When such ground-truth data is available, the tracking performance can be measured by tracking errors, which are computed

utilizing a root squared difference function between the ground-truth data $\mathbf{x}_{gt}$ and the tracking results $\mathbf{x}_{est}$:

$$error \triangleq \|\mathbf{x}_{est} - \mathbf{x}_{gt}\|_2 \tag{5.1}$$

Eq. (5.2) shows the average tracking error (ATE) measurement that we will utilize to evaluate tracking performance.

$$ATE = \frac{\sum_{t=1}^{T} error_t}{T}, \tag{5.2}$$

where t is the frame number and T is the total number of frames. However, in this paper, we are interested in measuring the accuracy and persistence of target tracking performance in a camera network. After a tracker loses track of the target object, the algorithm may present erratic behavior. Hence, it is difficult to show the tracking performance exclusively based on tracking errors, since the errors could indicate largely different values depending upon where the estimated tracks are lost. Here, we use an additional measurement to evaluate tracking performance. We measure to what extent a tracker tracks the target object successfully. Successful tracking is measured with a scoring function, which is a thresholding or decreasing function of distance between the target track and the ground-truth at the t^{th} frame. The average success score is defined in Eq. (5.3).

$$ATS = \frac{\sum_{t=1}^{T} score_t}{T}, \tag{5.3}$$

where t is the frame number, T is the total number of frames, and $score_t$ is a thresholding or decreasing function of the distance between the target track and the ground-truth at the frame t. Fig. 5.1 shows how the $score_t$ is assigned according to the Euclidean distance. When we use a thresholding score shown as the red dotted line, it assigns one if a tracking result is within a certain range; otherwise it assigns zero

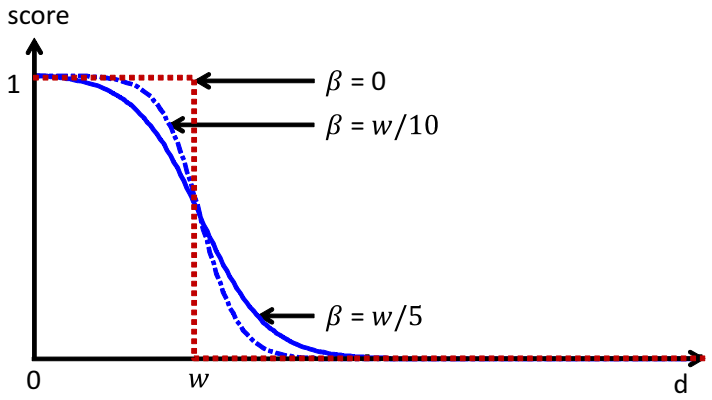


Figure 5.1.: Score function for tracking result evaluation.

to indicate tracking failure. When a decreasing function (blue lines) is utilized as a tracking score, it penalizes tracking results according to their accuracy. Hence, the tracking score can also show the accuracy and success of target tracking.

To evaluate our tests, we set the distance threshold as the target object width, w, and the scoring function as a sigmoid type function as in Eq. (5.4).

$$score \triangleq \frac{1+\exp(-w/\beta)}{1+\exp((d-w)/\beta)}, \tag{5.4}$$

where β is a control parameter for the transition width of the scoring function and d is the Euclidean distance between the estimated target position and the ground-truth at a given frame. In our experiments, we set β as $w/5$, as depicted in Fig. 5.1. As we can see, the thresholding approach is just a special case when $\beta = 0$.

5.1.2 Delivery Energy Efficiency

In lossy networks, transmitted data packets are frequently missed. The non-received packets also consume transmission energy, which is essentially wasted. If a tracker shows good tracking results with large packet loss (typically under the situa-

tion that it transmits a very large number of redundant packets), the tracking system requires high cost and may not be suitable for wireless systems. We measure the energy efficiency of delivered packets using *delivery energy efficiency* (DEE), which is defined as the non-wasted portion of the total transmission energy. Let us say e is the energy required to transmit a packet. Then, the total transmission energy E_{TX} is expressed as

$$E_{TX} = N_{TX} \times e, \tag{5.5}$$

where N_{TX} is the number of transmitted packets. Similarly, the amount of wasted energy E_{NR} is computed as

$$E_{NR} = N_{NR} \times e, \tag{5.6}$$

where N_{NR} is the number of lost packets. Then, DEE has the following form:

$$DEE \triangleq \frac{E_{TX} - E_{NR}}{E_{TX}} = \frac{N_{TX} - N_{NR}}{N_{TX}}. \tag{5.7}$$

We will utilize DEE as a measurement of energy efficiency for tracking data communication.

5.2 Simulations and Experiments in Wireless Camera Networks

In this section, simulated results using previously recorded image sequences are carried out to analyze the tracking performance of the proposed resource-aware method. Then, real-time experiments are shown on a WCN implementation.

5.2.1 Particle Filter Settings

In our experimental setup, we set the target state, $\mathbf{x}_t$, at time t as the object position (x_w, y_w) in the world coordinate plane:

$$\mathbf{x}_t = [x_w, y_w]^T. \tag{5.8}$$

For the state process model, we utilize a linear transition matrix corrupted by Gaussian noise:

$$\mathbf{x}_{t+1} = \boldsymbol{F}\mathbf{x}_t + \mathbf{u}_t, \tag{5.9}$$

where we set the identity matrix as the transition matrix, $\boldsymbol{F} = \boldsymbol{I}_{2\times 2}$, since, for simplicity, we do not accommodate object velocity in the state vector. The transition noise, $\mathbf{u}_t$, is set as a Gaussian noise model. Also, we consider $\mathbf{y}_t$ as the target object (or reference) feature at time t and $h_t(\mathbf{x}_t)$ as the extracted feature at a support point $\mathbf{x}_t$ in the image at time t. In our implementation, we use color histogram features as described in [16, 18, 59]. We compute the object histogram at the object position on the image coordinate, which is transformed from the object position on the world coordinate frame by the precomputed homography obtained from the camera calibration information. The likelihood probability is set to

$$p(\mathbf{y}_{i,t}|\mathbf{x}_t) = \frac{1}{Z}\exp\left(-\frac{d(h_{i,t}(H_i\mathbf{x}_t), \mathbf{y}_{i,t})}{\lambda}\right), \tag{5.10}$$

where Z is the normalization constant, H_i is the homography between the world coordinate ground plane and the i^{th} camera image plane, λ is the observation noise power, and $d(x, y)$ is the Euclidean distance function between x and y. Note that the distance can be computed instead with the Bhattacharyya distance utilized in [16, 18] as shown in [22] or with any other suitable distance metric. The color histograms are

Table 5.1: Data payload in a packet.

	1 packet (12 bytes)
Synch	6 particle weights
GMM	1 set of GMM parameters
Parzen	3 Parzen samples

computed in the RGB color space with 38 bins in each dimension. The noise level (power) was set as a fixed constant, $\lambda = 0.06$.

5.2.2 Communication Packet Payloads and Packet Loss Models

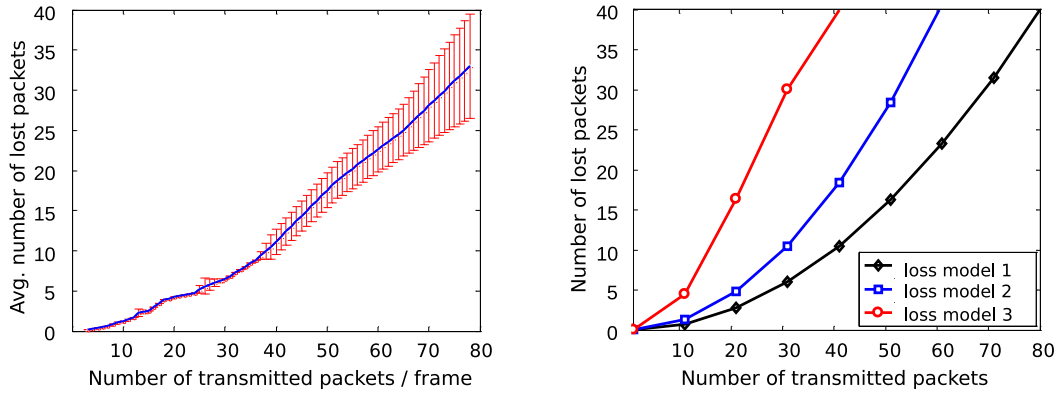
In our implementation, we describe a floating point variable such as a particle weight with 2-byte precision. For communications, we transmit 12 bytes of payload in a packet. For the synchronized particle filter, 6 particle weights are loaded in a packet, as a particle weight requires 2 bytes to be described. In the Parzen particle filter, 4 bytes are needed to describe a Parzen sample composed of x and y positions, hence 3 Parzen samples are loaded in a packet. A set of GMM mixture parameters consists of 6 floating point variables, which are a GMM component weight, x and y positional means, and the variances and covariance between x and y. This set of GMM parameters requires 12 bytes, and hence in the GMM particle filter, a single set of GMM parameters is loaded in a packet. Table 5.1 shows the summary of the different data payloads in a packet.

We set the number of particles $K = 300$ for the probability conversion based particle filter methods. However, for the synchronized particle filter, the number of particles is determined by the number of packets available to a local node. For example, when 10 packets are assigned to a node for particle data communication, the number of particles in the synchronized particle filter is 60 (10 multiplied by 6).

To obtain a packet loss model, the clustering [7] and the Parzen based tracking algorithms with a sensing rate of 0.5 seconds were executed under the Avrora (AVR) simulator [60] with the CSMA (carrier sense multiple access with collision avoidance) protocol [53]. We varied the numbers of packets per frame assigned to each node as well as the number of cluster members in each cluster (2 to 7 cluster members, including cluster head) so that the total number of packets transmitted per frame varied between approximately 0 and 80. Fig. 5.2 (a) shows the average packet loss of the simulations, which were repeated 30 times on each combination of number of packets per frame and cluster members. Based on the simulation results, we created three packet loss models (shown in Fig. 5.2 (b)) to investigate the performance of the proposed resource-aware approach. These models are supposed to represent severe, moderate, and mild communication failure scenarios in our experimental evaluation in the following sub-sections.

5.2.3 Simulated Experiments

We tested three multiple image sequence sets in our simulations; Campus (3 image sequences) [61], PETS (4 image sequences) [62], and Lab (8 image sequences) image sets. Examples of the publicly available Campus and PETS image sequences, which are captured in outdoor environments, are shown in Fig. 5.3 (a) and (b), respectively. Fig. 5.3 (c) shows examples of the Lab image sequences that are captured in an indoor environment. In each image sequence set, the red boxes indicate the images captured by the cluster heads. The target objects in the image sequences are marked by a small yellow box. We select the test image sequences in which all cameras have common object views so that all the cameras take part in the tracking process, one as the cluster head and the others as cluster members. We assume that all cameras are located in single hop communication range to the cluster head. All of the tests



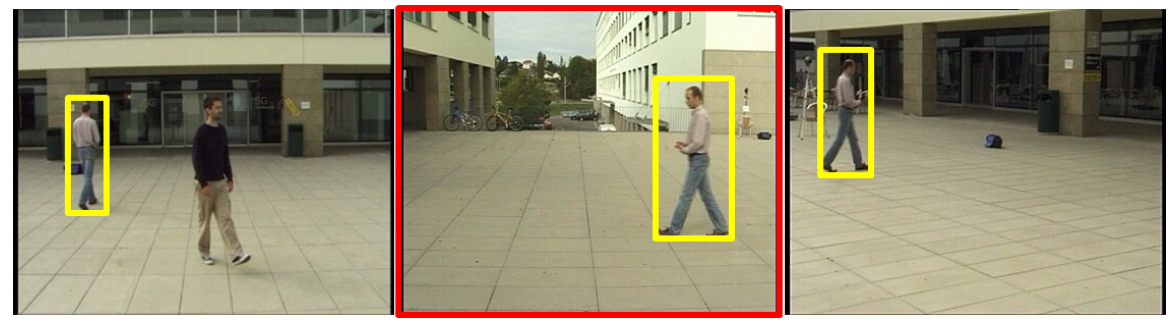
(a) (b)

Figure 5.2.: Average packet loss under AVR simulator (a) and packet loss models (b). (a) Red bars indicate standard deviations of the packet loss. (b) The black line model is obtained from the AVR simulation and the red and blue lines are set for a moderate and a severe loss case for the resource-aware tracking test, respectively. In the packet loss model 3, as the number of transmitted packets increases, most transmitted packets are lost.

were done under the same transition dynamic model and observation noise level. The experiments were carried out using two different target packet loss rates, $M = 0.1$ and $M = 0.2$, for the resource-aware distributed particle filters under the three loss models described in Fig. 5.2 (b). For the non resource-aware distributed particle filters, 4 to 10 packet loading scenarios were tested under the same two loss models. In these experiments, ATSs and ATEs are computed by setting the tracking results of a centralized particle filter as the ground-truth.

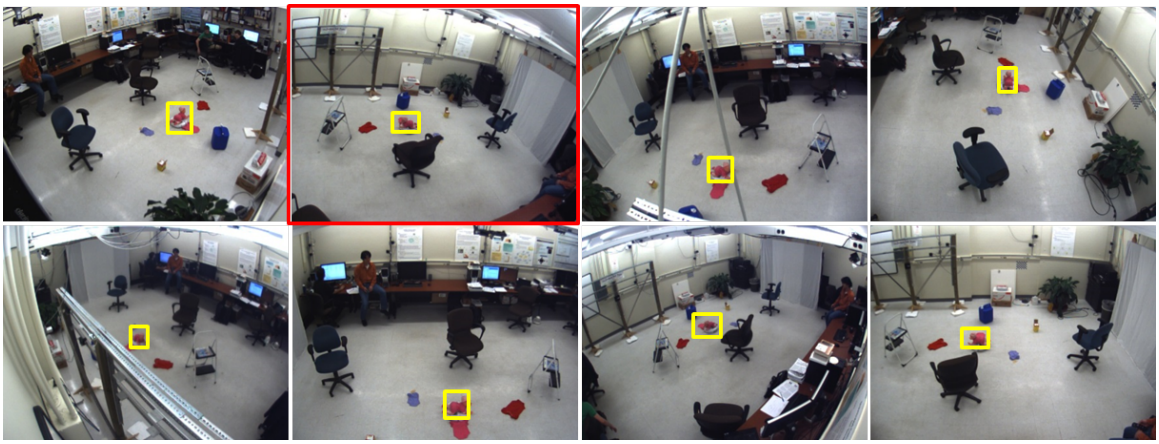
We first looked at the performance of the standard distributed particle filters, in which the resource-aware method is not employed. Fig. 5.4 shows the average tracking scores and the average tracking errors for the Campus, PETS, and Lab sequences under lossless packet communication. All three distributed particle filters, the synchronized, the GMM, and the Parzen approaches, show reliable tracking performances. As the figure shows, all ATSs are close to 1 and the ATEs are within few centimeters, since the probability representations are not subject to any degradation caused by dropped packets. As the number of assigned transmission packets are increased, the tracking performance does improve but not noticeably (ATSs are increased and ATEs are decreased) as shown in Fig. 5.4. Note that the three different approximations of the object probability distribution cause a marginal difference in error values.

However, when packet losses occur, the tracking performances of the distributed particle filters are degraded. Figs. 5.5 and 5.6 show the average tracking scores and the average tracking errors for the Campus, PETS, and Lab sequences under loss model 1, 2, and 3. Figs. 5.7 and 5.8 depict the corresponding delivery energy efficiencies and transmitted and received packets. As the number of transmitted packets in the network increases (from 4 to 10 packets), the number of lost packets also increases as shown in Fig. 5.8. The synchronized particle filter shows that it



(a)

(b)



(c)

Figure 5.3.: Test image sequences. (a) Campus, (b) PETS and (c) Lab sequences. The red image box indicates the image captured at the cluster head; the small yellow boxes in all images show the target object.

is extremely vulnerable to packet losses; the tracking performance degrades rapidly as the number of lost packets increases (see Figs. 5.5 and 5.6). In the synchronized particle filter, the collaborative processing, which consists of computing the joint probability, requires particle synchronization. When data losses occur, the particle weights in the lost support points become unavailable, which leads to a distorted joint probability and incorrect object track estimation. On the other hand, the GMM and Parzen particle filters show robust performances to lossy data communication. In the Campus and PETS sequence experiments, which utilized 3 and 4 cluster members, the GMM and Parzen tracking algorithms present around 90 % tracking success rate (0.9 of ATS) under the packet loss model 1 and 2, in which the packet loss rates are less than 30 %. However, increasing the packet loss rate, which happens in the Lab sequence experiments with the packet loss models 1, 2, and 3 and the PETS sequence experiments with the packet loss model 3, the GMM particle filter shows degraded performances as shown in Fig. 5.5 (c), (f), (h), and (i), as data traffic increases as shown in Fig. 5.8 (c), (f), (h), and (i). The Parzen particle filter shows additional robustness with respect to the GMM approach in at least one of these three severe scenarios (plots (c) and (f)).

When the resource-aware method is applied to the distributed particle filters, it confines the packet loss rate to a pre-assigned level M_{max}. For the synchronized particle filter, the resource-aware method does not improve the tracking performance much, as it does not guarantee lossless packet transmission but rather a certain rate of packet loss, and the synchronized particle filter cannot operate properly even under modest communication failures. However, the resource-aware method drastically improves the tracking performances of the GMM and the Parzen distributed particle filters as shown in Figs. 5.5 and 5.6, and increases the packet delivery energy efficiencies as shown in Fig. 5.7. For the most severe packet loss case, shown in Fig.

5.8 (i), the resource-aware method still preserves the requested packet loss rate and improves the tracking performances of the GMM particle filter from a mean ATS of 0.11 to an ATS of 0.52 and that of the Parzen particle filter from a mean ATS of 0.04 to an ATS of 0.74, as shown in Fig. 5.5 (i) (the mean ATS of the non-resource aware particle filters is computed by averaging the ATSs for 4 to 10 packet transmissions).

Selected object trajectories under lossless packet communication and loss model 1, 2, and 3 are shown in Figs. 5.9, 5.10, 5.11, and 5.12, respectively. Note that in the trajectory plots, the world plane is represented in the x and y axes, and the frame number is indicated in the z axis. As previously shown, high tracking errors even in the presence of modest communication failures. Therefore, the object trajectories estimated by the synchronized particle filter are far deviated from the ground-truth, and hence, to better visualize the performance of the resource-aware method, we chose to plot only the trajectories of the GMM and the Parzen particle filters in the remaining trajectory plots. Figs. 5.11 (third column) and 5.12 (second and third columns) show that the estimated trajectories become significantly more stable when the resource-aware method is applied to the GMM and the Parzen particle filters, confirming that the resource-aware method improves the object tracking performance, as previously shown in Fig. 5.5 (c), (h), and (i). For the most severe packet loss case (the Lab sequence test with the loss model 3), since almost no data is received, all the distributed particle filters lose track of the object, as shown in Fig. 5.12 (b) and (c). The resource-aware method reduces the number of packets to be transmitted in this case and it allows a certain portion of packets to be communicated (as shown in Fig. 5.8 (i)), thereby restoring to a great extent the functionality of the GMM particle filter and allowing the Parzen particle filter to operate almost perfectly, as shown in the second and third columns of Fig. 5.12. Snapshots of the tracking results are shown in Fig. 5.13.

To show a more concise and integrative view of the distributed particle filter performances, we introduce a new metric that combines the ATS and DEE metrics. Fig. 5.14 shows the combined measure, which is computed by multiplying the ATS and DEE. As shown in Fig. 5.14 (a), the distributed particle filters do not take advantage of utilizing the resource-aware method when packet loss is negligible. However, the resource-aware methods shows improved performances for the GMM and Parzen particle filters when packet loss is higher as shown in Fig. 5.14. Although the resource-aware methods tend to present slightly better performance when $M_{max} = 0.1$, when packet loss is severe as in the case of the particle filters with 8 cluster members running under the loss model 3, the performances of the two resource-aware methods (and $M_{max} = 0.2$) does not show noticeable difference as shown in Fig. 5.14 (i), since the best packet loss rate that can be achieved by either resource-aware method is limited to the minimal packet loss condition. In this experiment, the achieved packet loss rates of both resource-aware methods using $M_{max} = 0.1$ and $M_{max} = 0.2$ were approximately 0.28 packet loss rate (i.e., $M = 0.28$). Note that in the other experiments the resource-aware method for the GMM and Parzen particle filters with 0.2 maximum loss rate ($M_{max} = 0.2$) show lower values than with 0.1 maximum loss rate ($M_{max} = 0.1$) in the combined measure because although their ATSs are comparable there is a difference of 0.1 between the two maximum loss rates

To understand how the performances of the particle filters are affected by the number of nodes in the network and by the amount of traffic generated by each node, we consider scenarios in which clusters consisted of 3 to 8 members and the number of packets generated by each cluster member varied from 4 to 10. For these experiments, we used only the Lab sequences as they correspond to the network with the largest number of cameras and the greatest amount of overlap between camera views. Fig. 5.15 shows the delivery energy efficiencies and received packets. Note

that packet losses and the packet control mechanism of the resource-aware method are independent of the particle filters. Hence, we show common DEEs and RX packets for all the distributed particle filters in Fig. 5.15. The resource-aware method preserves the DEEs of the distributed particle filters by monitoring the rate of lost packets and adjusting the number of data packets, whereas non-resource-aware particle filters have degraded DEEs as they do not have any adaptive mechanism for lossy communication environments.

Figs. 5.17 and 5.18 depict the tracking performances with ATSs and ATEs. As the figures indicate, the resource-aware methods perform significantly better than the non-resource aware particle filters in the majority of the scenarios, especially when the number of cluster members is higher (and consequently the network traffic is heavier).

Again, the synchronized particle filter cannot benefit much from the resource-aware method. The experiments shown in Fig. 5.17 (a), (d), and (g) demonstrate that the tracking performance is dependent on the number of packet losses as shown in Fig. 5.15 (d) - (f). The resource-aware method does not help the performance much, since it keeps a certain level of packet loss rate as shown in Fig. 5.15.

The GMM components are weighted as described in Section 3.3.1.2. Therefore, data packets of the distributed GMM particle filter are not equally important, since a packet consists of a mixture weight. When highly weighted packets are lost in data communication, the converted probability from the received packets represents the transmitted probability poorly, which causes unreliable tracking. The experimental results in Figs. 5.17 and 5.18 show a tendency that higher packet loss rates caused by increasing the number of cluster members and packet transmissions degrade the tracking performance of the GMM particle filter. This tendency is explained by the fact that higher loss rates increase the odds of dropping important packets. The

resource-aware method confines the packet loss rate, which reduces the chance of missing highly weighted packets. Hence, the performance of the GMM particle filter is improved with the resource-aware method under lossy communication environments. The GMM particle filter shows better tracking performance with a smaller maximum missing rate ($M_{max} = 0.1$ in the experiments).

In the Parzen distributed particle filter, Parzen samples are packed into transmission packets. Communication failures cause packets to be dropped at random, which is essentially equivalent to reducing the resolution of Parzen sampling since Parzen samples are generated by a random selection from the equally weighted particles. Furthermore, Parzen samples are distributed densely at the support points that have high particle weights. When the resolution of the Parzen representation is not at a critical level (i.e., the number of received Parzen samples is not too small) to represent an object probability, the Parzen particle filter can provide reliable tracking performance. Hence, the Parzen particle filter is less vulnerable to packet losses than the synchronized and the GMM particle filters, as our experiments demonstrate. In terms of tacking accuracy, the resource-aware method does not improve the performance of the Parzen particle filter much as shown in Fig. 5.17 (c) and (f) if packet loss rate is not critical as shown in Fig. 5.15 (d) and (c). Note that the resource-aware method enables even a small number of packet communications in a extremely lossy environment as shown in Fig. 5.15 (f), and the Parzen particle filter takes advantage of the resource-aware method in this case. In general, even for lower packet loss rates, the resource-aware method significantly increases the DEE without degrading the tracking performance of the Parzen particle filter, as shown in Fig. 5.15 (a) - (c).

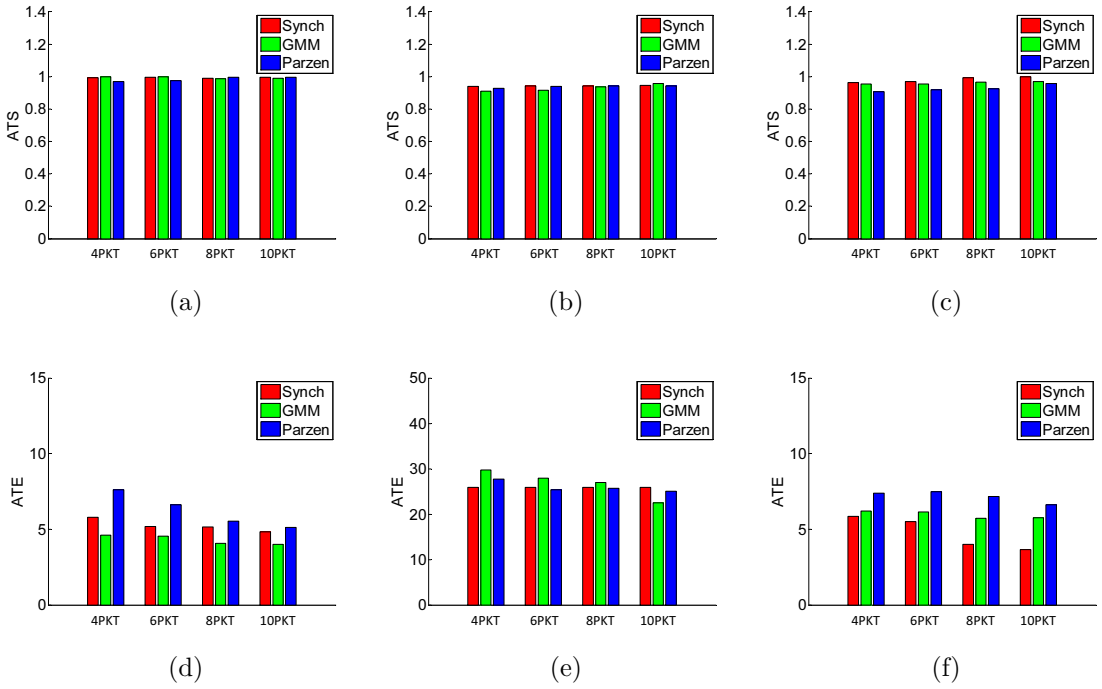


Figure 5.4.: Average tracking scores (ATSs) and average tracking errors (ATEs) under lossless packet communication. (a) - (c): ATS for Campus, PETS, and Lab sequences. (d) - (f): ATE for Campus, PETS, and Lab sequences.

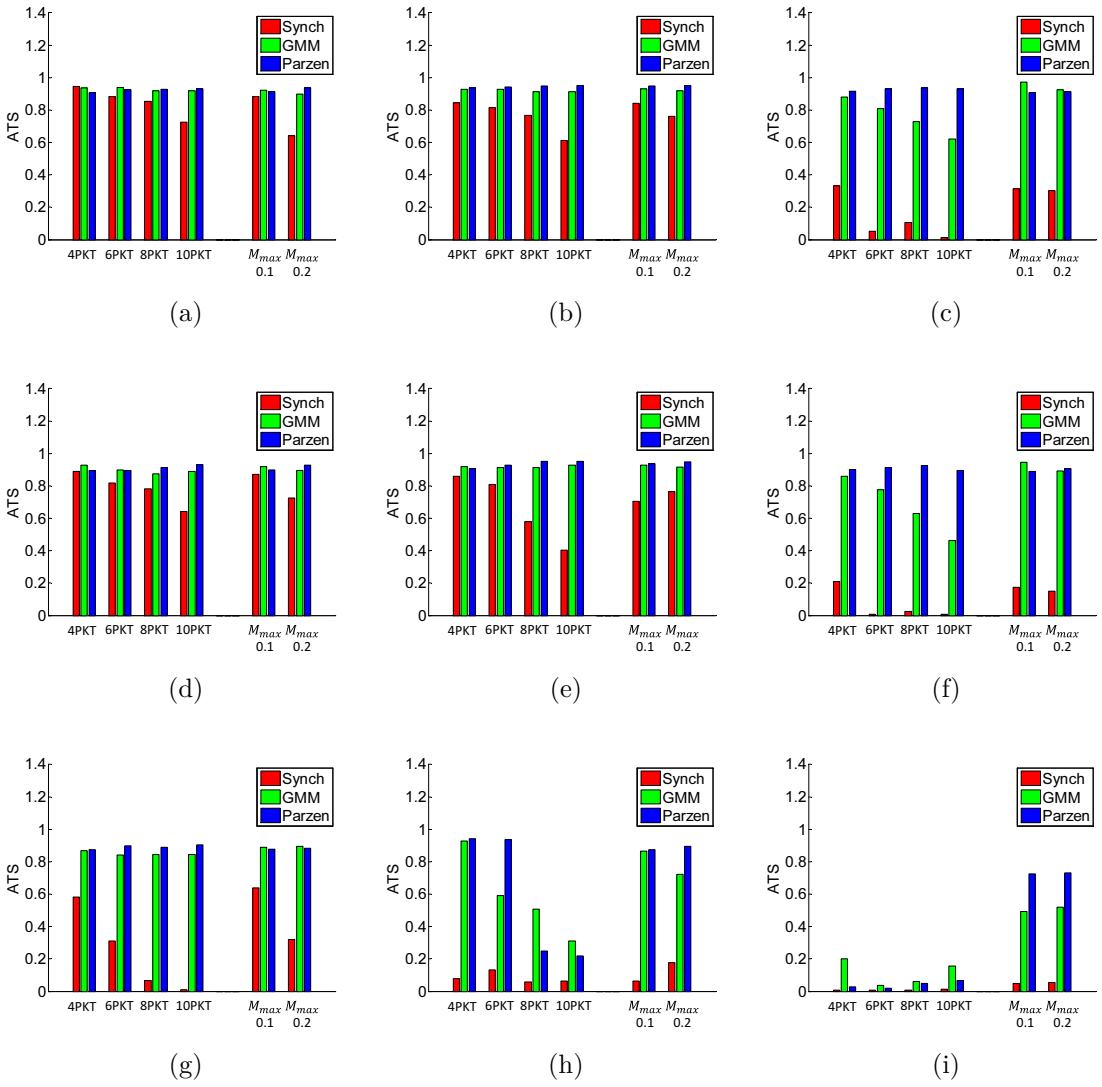


Figure 5.5.: Average tracking scores (ATSs) for the three test sequences. From left to right, each column shows ATSs for Campus, PETS, and Lab sequences, respectively. From top to bottom, each row depicts ATSs under loss model 1, 2, and 3, respectively.

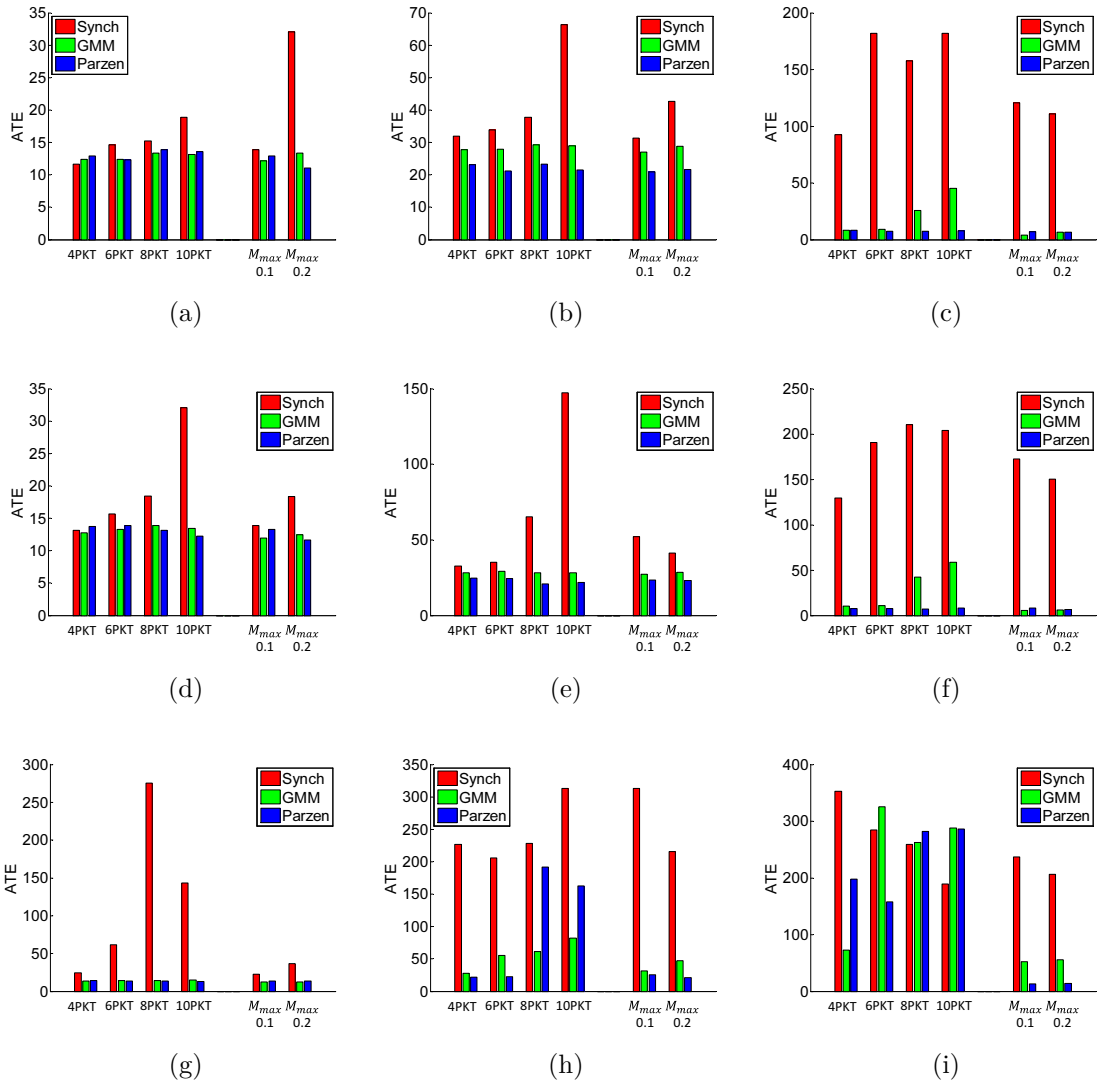


Figure 5.6.: Average tracking errors (ATEs) for the three test sequences. From left to right, each column shows ATEs for Campus, PETS, and Lab sequences, respectively. From top to bottom, each row depicts ATEs under loss model 1, 2, and 3, respectively.

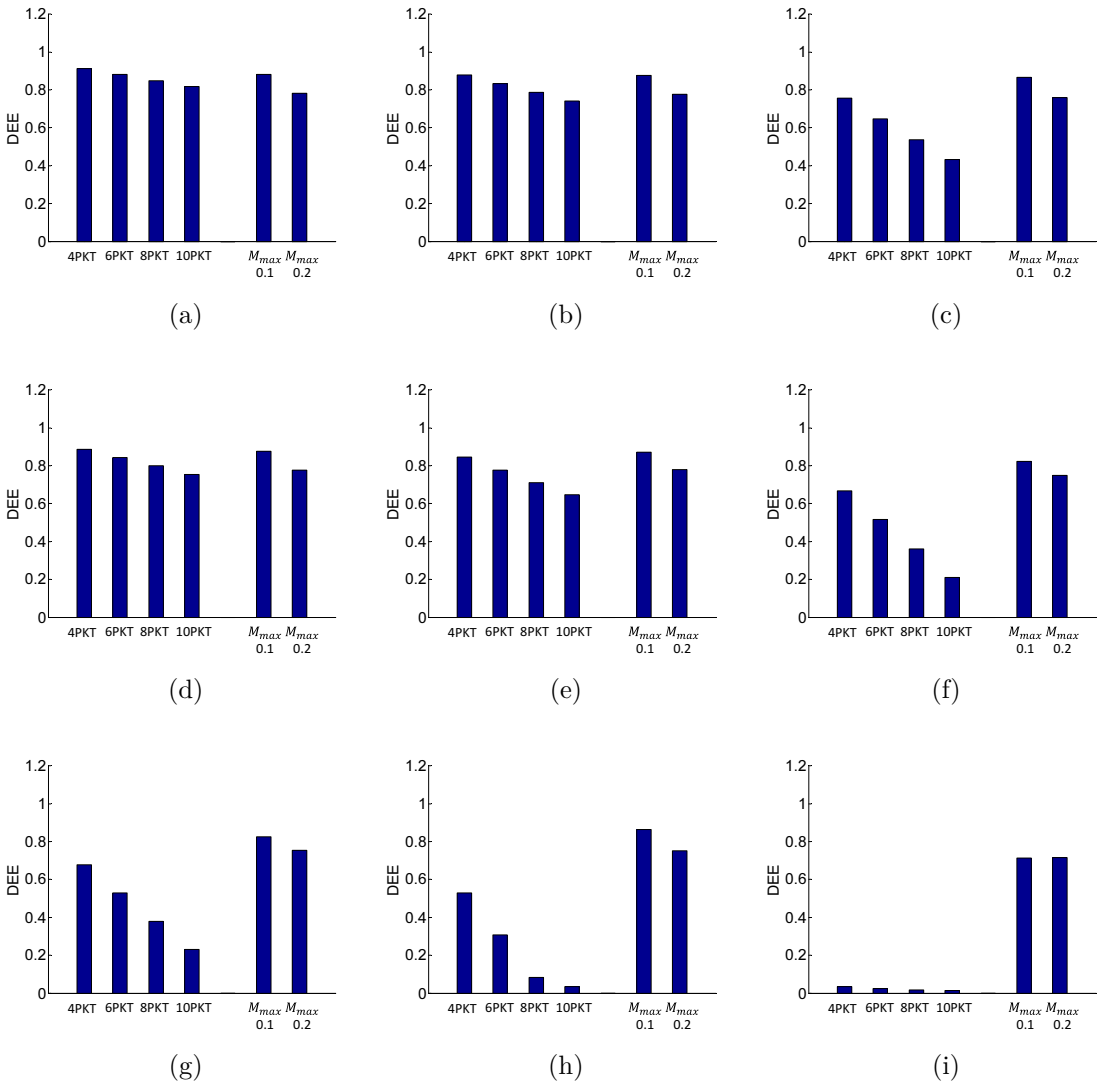


Figure 5.7.: Delivery energy efficiencies (DEEs) for the three test sequences. From left to right, each column shows DEEs for Campus, PETS, and Lab sequences, respectively. From top to bottom, each row depicts DEEs under loss model 1, 2, and 3, respectively.

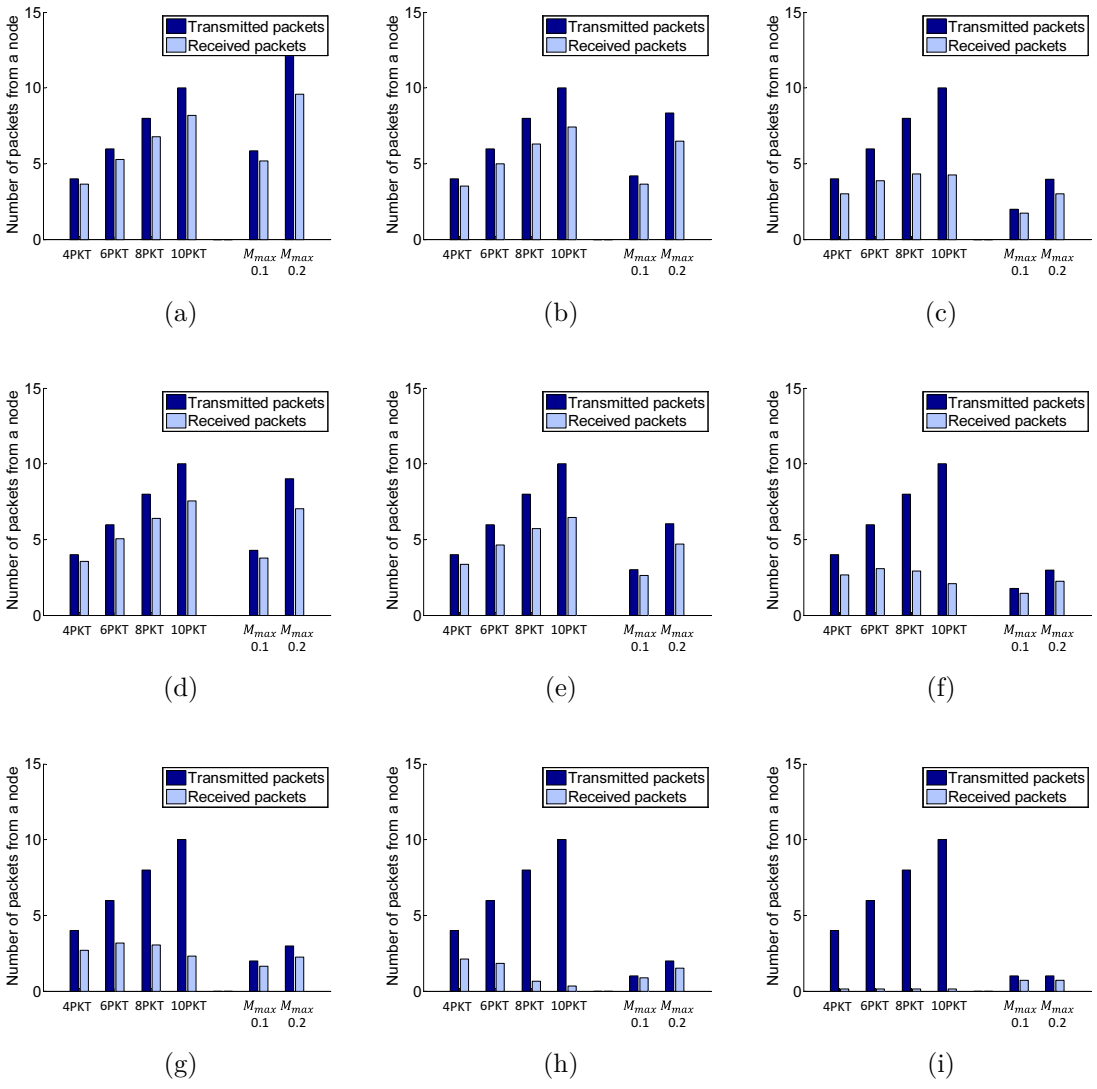


Figure 5.8.: TX and RX packets for the three test sequences. From left to right, each column shows TX and RX packets for Campus, PETS, and Lab sequences, respectively. From top to bottom, each row depicts TX and RX packets under loss model 1, 2, and 3, respectively.

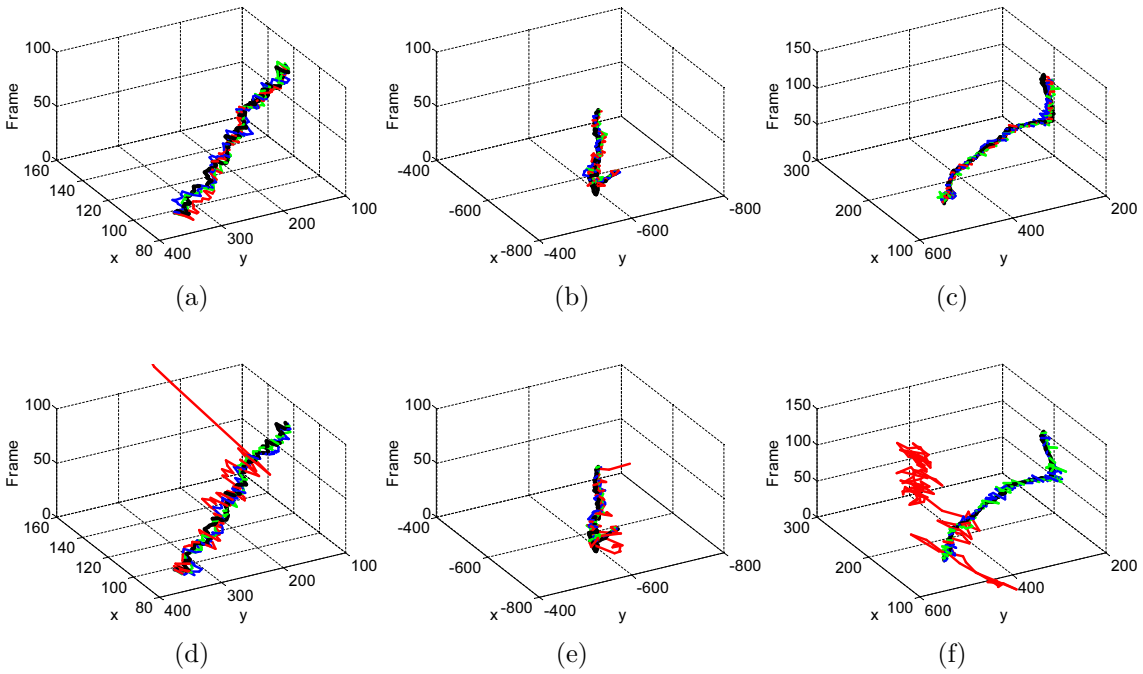


Figure 5.9.: Trajectories of estimated object positions (unit: cm). (a) - (c): Trajectories of non resource-aware methods when 8 packets per node were transmitted for the Campus, PETS, and Lab sequences under lossless packet communication. (d) - (f): Trajectories of resource-aware methods when $M_{max} = 0.2$ for Campus, PETS, and Lab sequences under the loss model 1. The red, blue, green, and black lines indicate the object trajectories estimated by the synchronized, Parzen, GMM, and centralized particle filters, respectively.

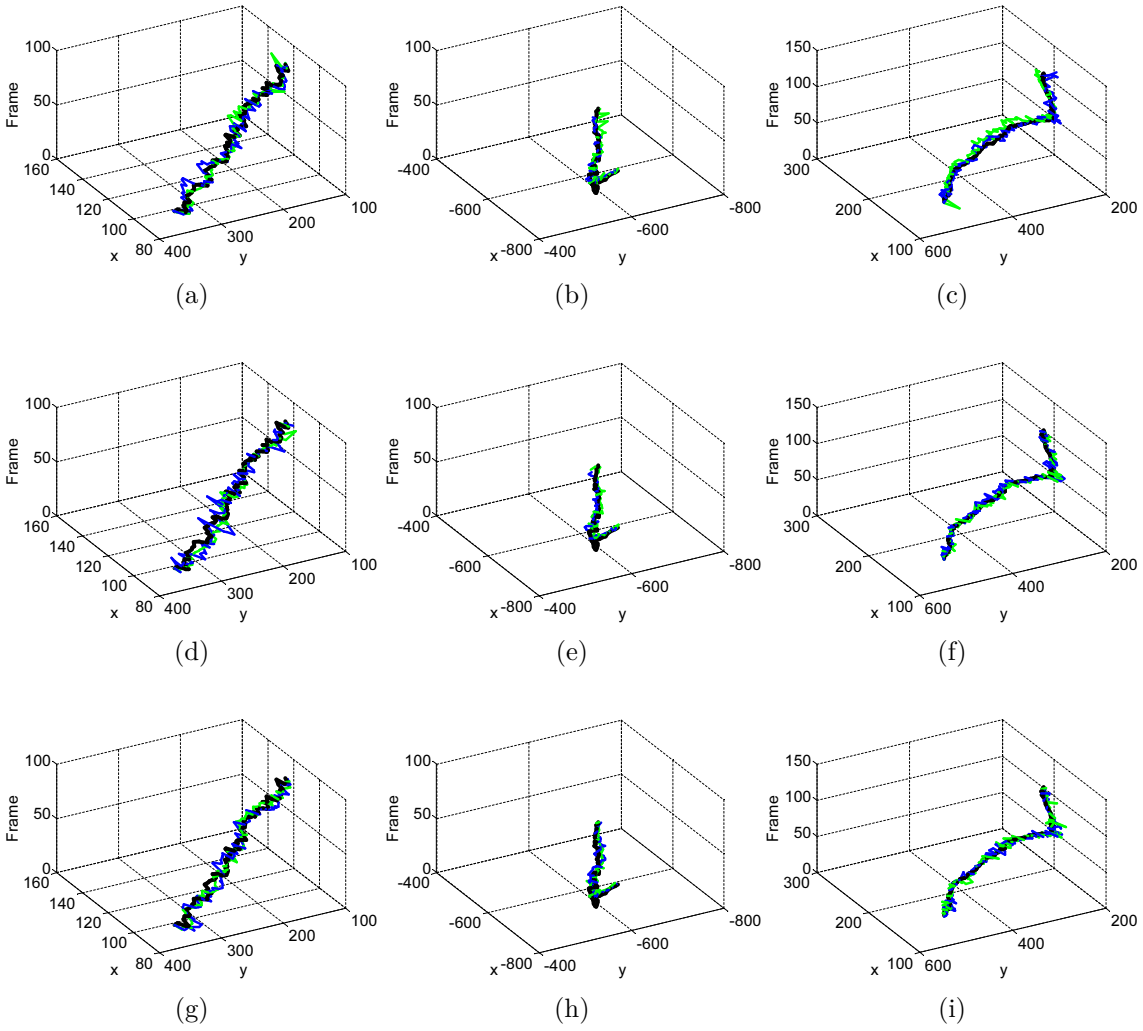


Figure 5.10.: Trajectories of estimated object positions under the loss model 1 (unit: cm). (a) - (c): Trajectories of non resource-aware methods when 8 packets per node were transmitted for the Campus, PETS, and Lab sequences. (d) - (f): Trajectories of resource-aware methods when $M_{max} = 0.1$ for Campus, PETS, and Lab sequences. (g) - (i): Trajectories of resource-aware methods when $M_{max} = 0.2$ for Campus, PETS, and Lab sequences. The blue, green, and black lines indicate the object trajectories estimated by Parzen, GMM, and centralized particle filters, respectively.

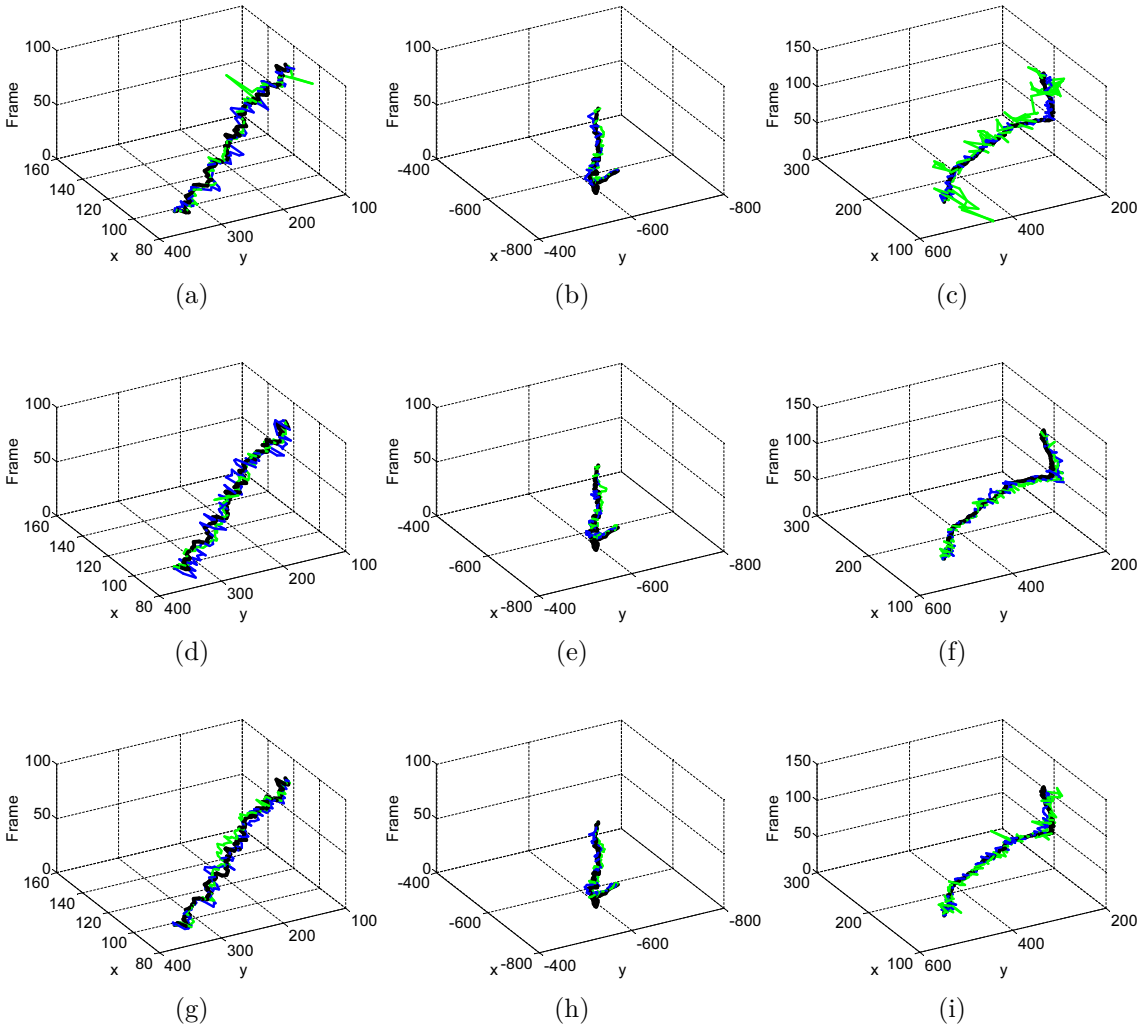


Figure 5.11.: Trajectories of estimated object positions under the loss model 2 (unit: cm). (a) - (c): Trajectories of non resource-aware methods when 8 packets per node were transmitted for the Campus, PETS, and Lab sequences. (d) - (f): Trajectories of resource-aware methods when $M_{max} = 0.1$ for Campus, PETS, and Lab sequences. (g) - (i): Trajectories of resource-aware methods when $M_{max} = 0.2$ for Campus, PETS, and Lab sequences. The blue, green, and black lines indicate the object trajectories estimated by Parzen, GMM, and centralized particle filters, respectively.

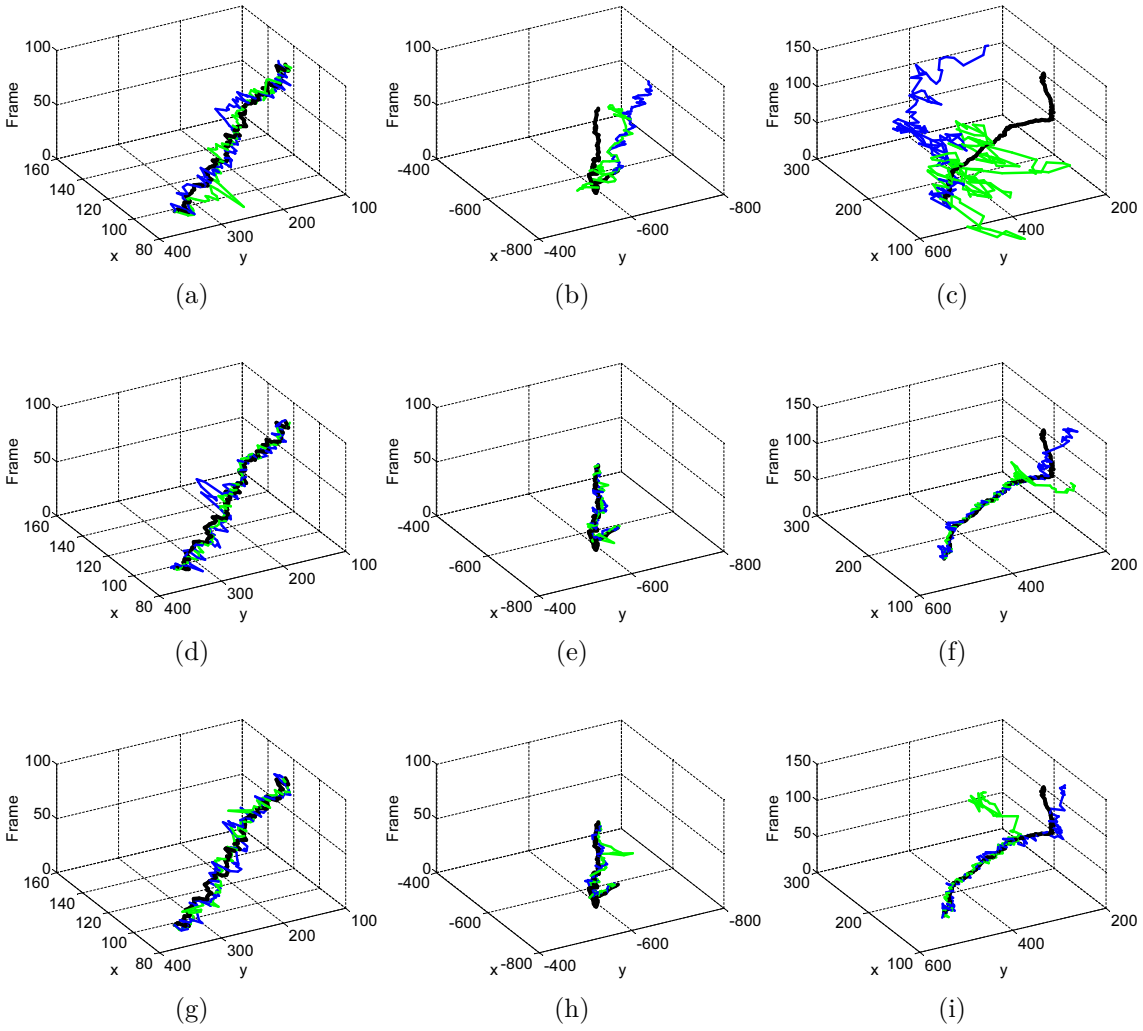
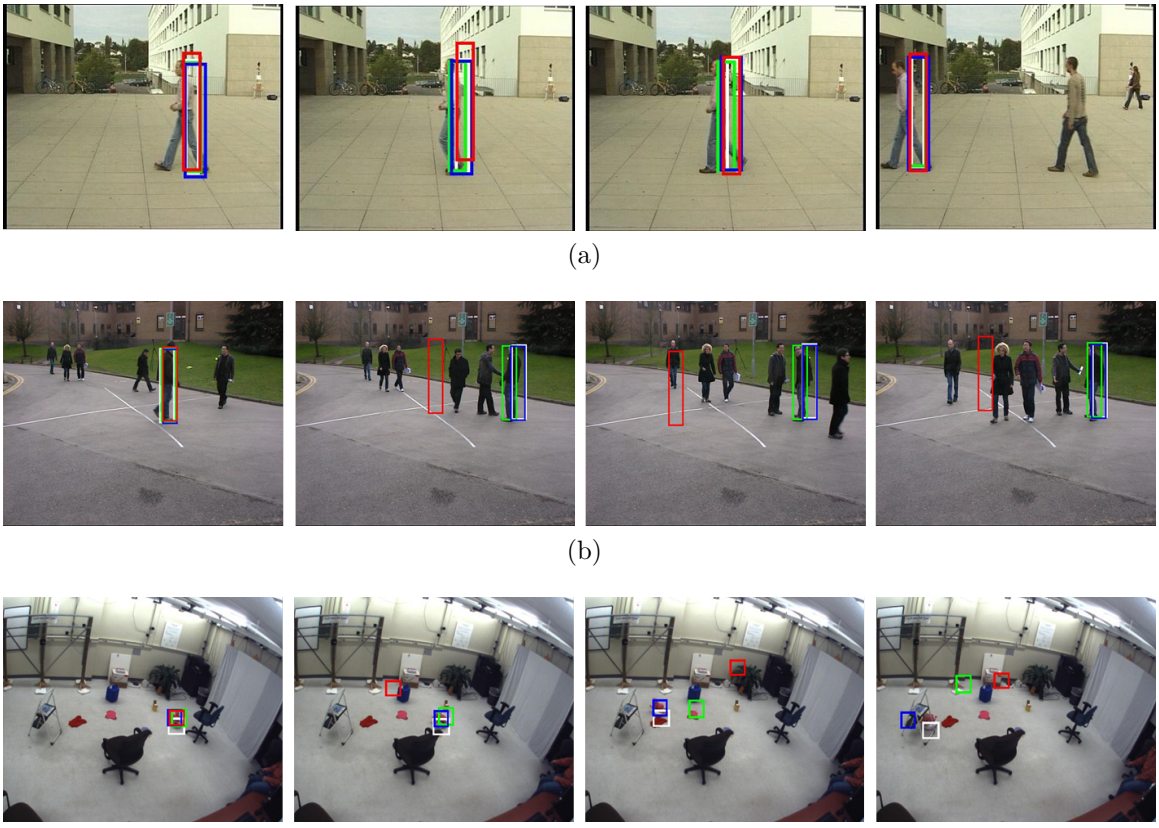


Figure 5.12.: Trajectories of estimated object positions under the loss model 3 (unit: cm). (a) - (c): Trajectories of non resource-aware methods when 8 packets per node were transmitted for the Campus, PETS, and Lab sequences. (d) - (f): Trajectories of resource-aware methods when $M_{max} = 0.1$ for Campus, PETS, and Lab sequences. (g) - (i): Trajectories of resource-aware methods when $M_{max} = 0.2$ for Campus, PETS, and Lab sequences. The blue, green, and black lines indicate the object trajectories estimated by Parzen, GMM, and centralized particle filters, respectively.



(a)

(b)

(c)

Figure 5.13.: Tracking results of (a) Campus, (b) PETS, and (c) Lab sequences using resource-aware method set with $M_{max} = 0.1$ under the loss model 3. The red, green, blue, and white boxes indicate the object tracks estimated by synchronized, GMM, Parzen, and centralized particle filters, respectively. The tracking results were captured at the cluster head.

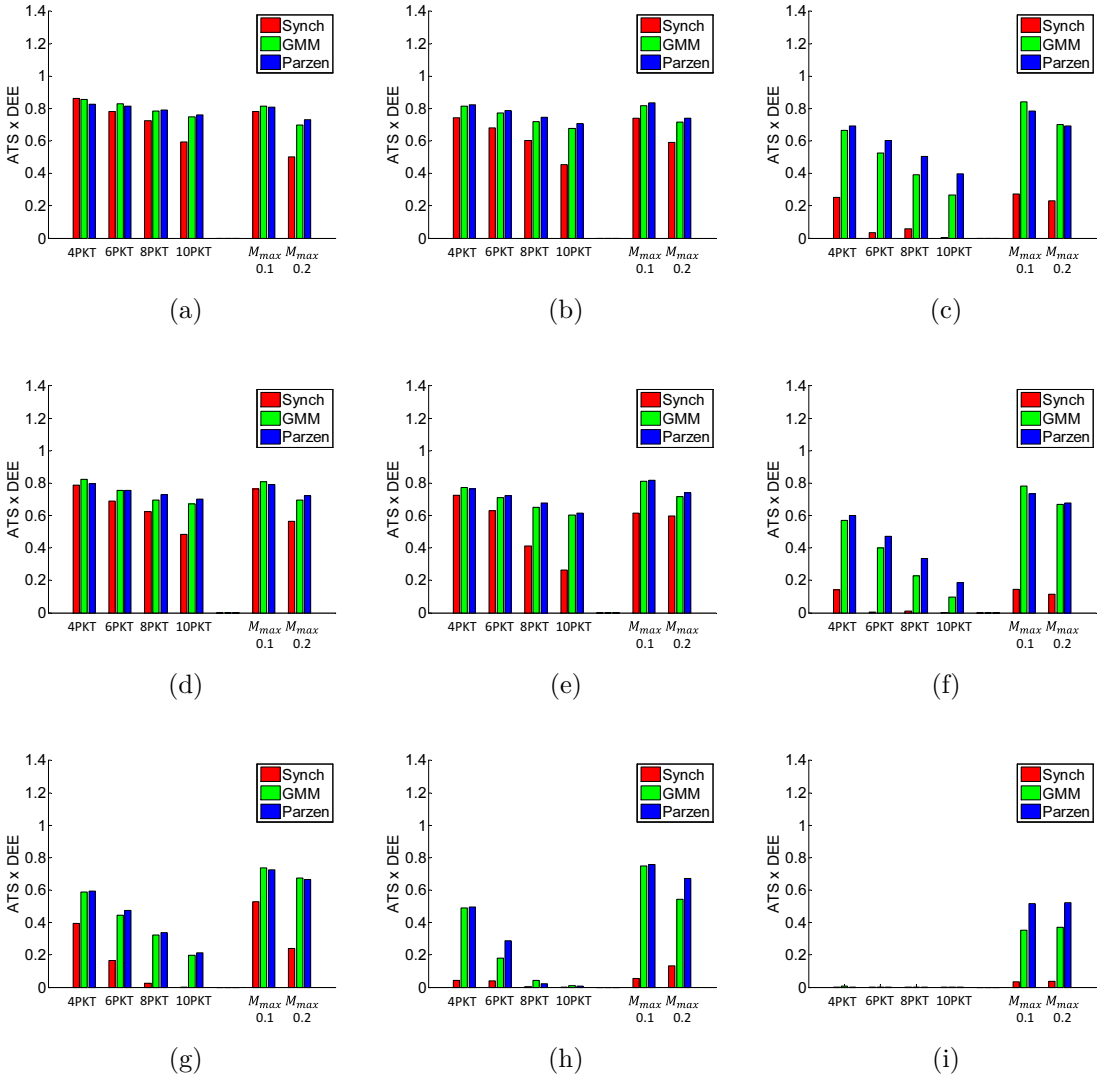


Figure 5.14.: Average tracking scores (ATSs) with delivery energy efficiency (DEE) for the three test sequences. From left to right, each column shows ATS×DEEs for Campus, and PETS, and Lab sequences, respectively. From top to bottom, each row depicts ATS×DEEs under loss model 1, 2, and 3, respectively.

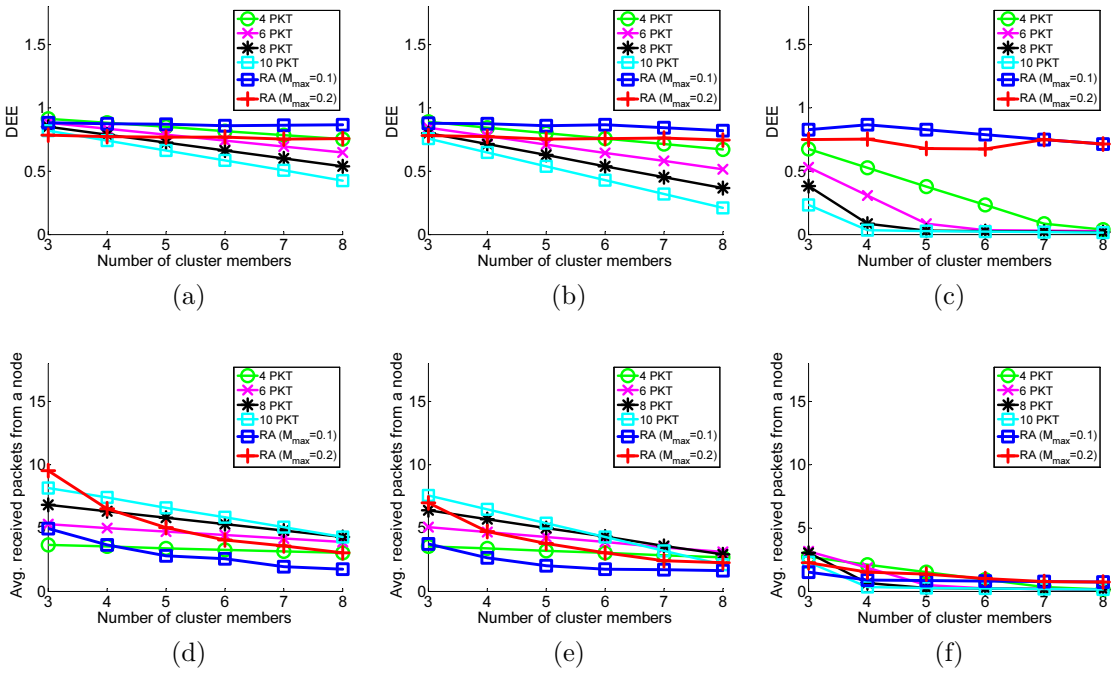


Figure 5.15.: Delivery energy efficiencies (DEEs) and RX packets. (a) - (c): DEEs of 3 - 8 Lab sequences under loss model 1, 2, and 3, respectively. (d) - (f): RX packets for 3 - 8 Lab sequences under loss model 1, 2, and 3, respectively.

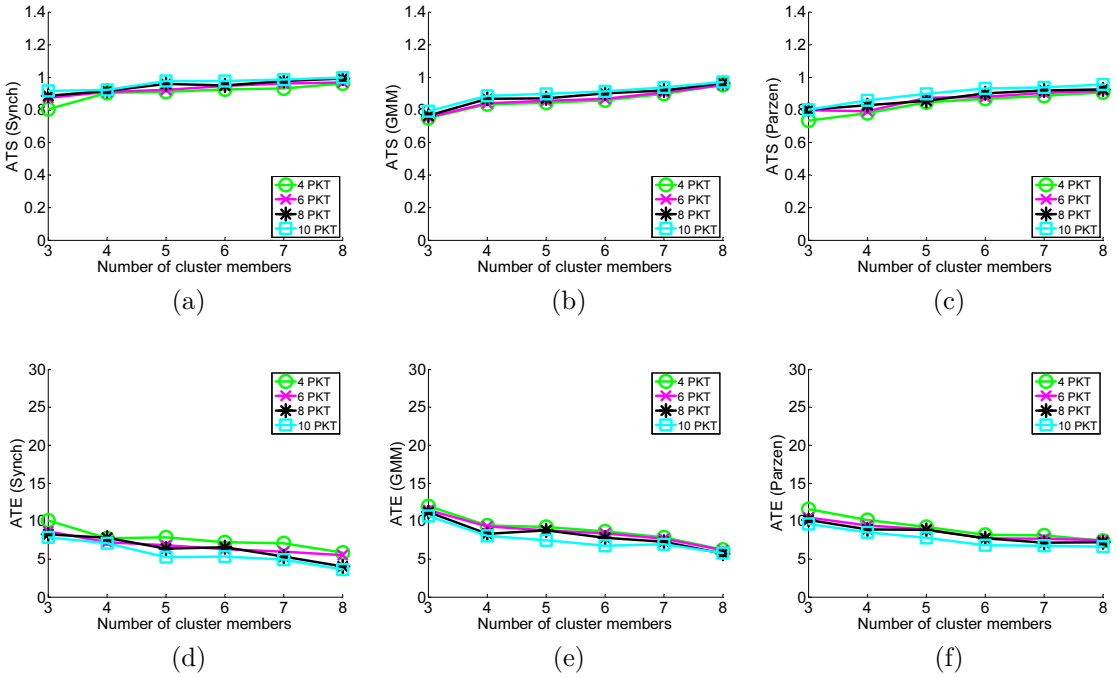


Figure 5.16.: Average tracking scores (ATSs) and average tracking errors (ATEs) of 3 - 8 Lab sequences under the lossless packet communication. (a) - (c): ATS for the synchronized, GMM, and Parzen particle filters. (d) - (f): ATE for for the synchronized, GMM, and Parzen particle filters.

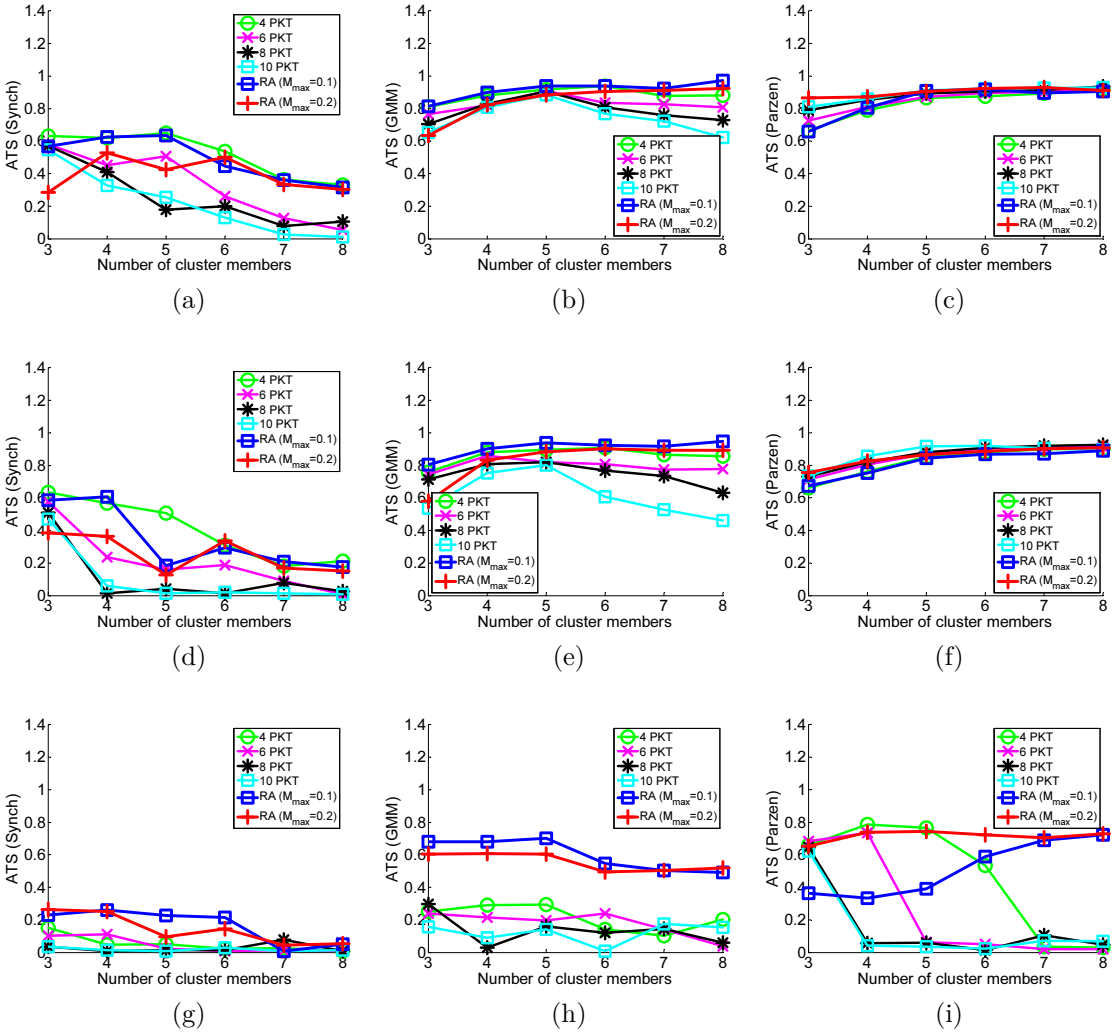


Figure 5.17.: Average tracking scores (ATSs) of 3 - 8 Lab sequences. From left to right, each column shows ATSs for the synchronized, GMM, and Parzen particle filters, respectively. From top to bottom, each row depicts ATSs under loss model 1, 2, and 3, respectively.

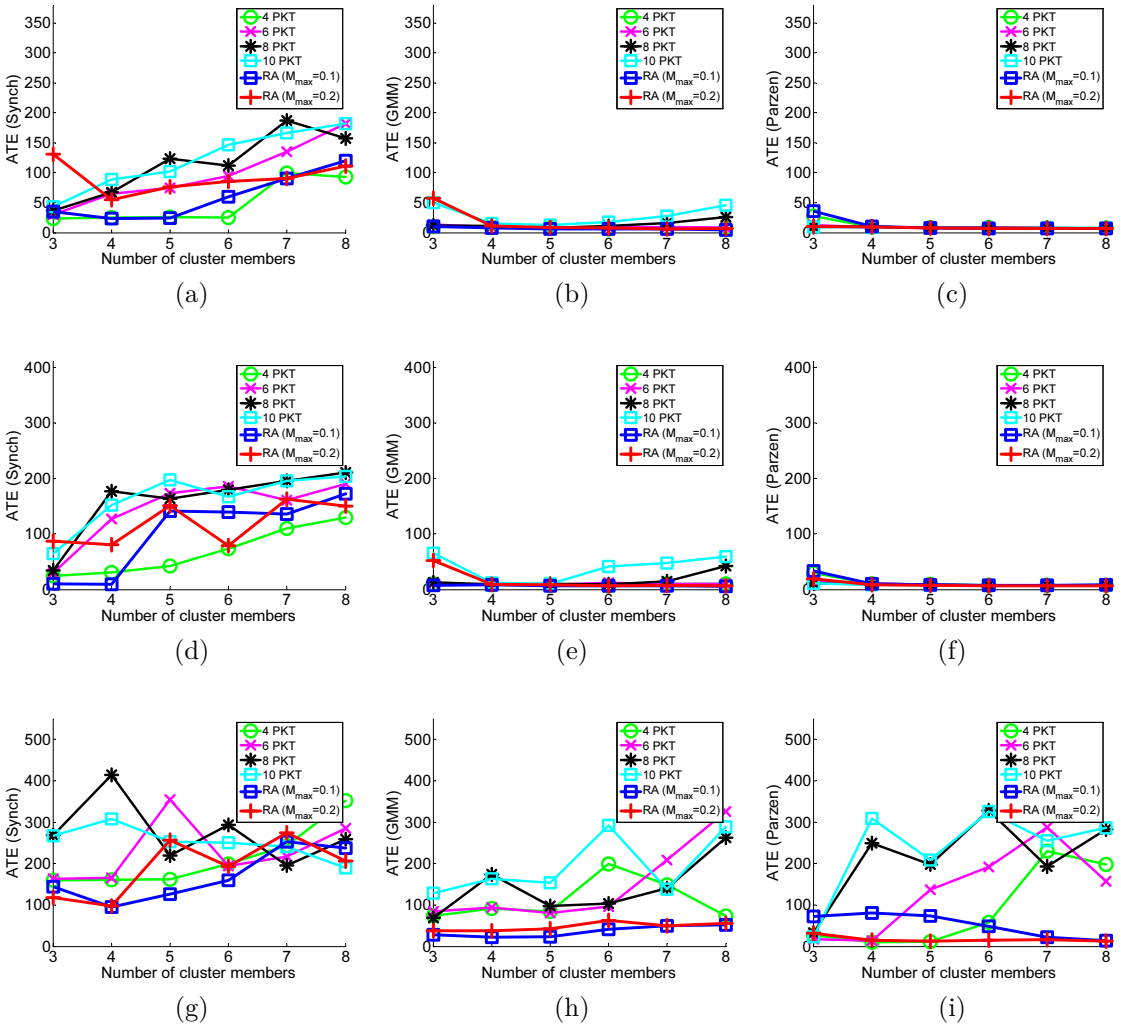
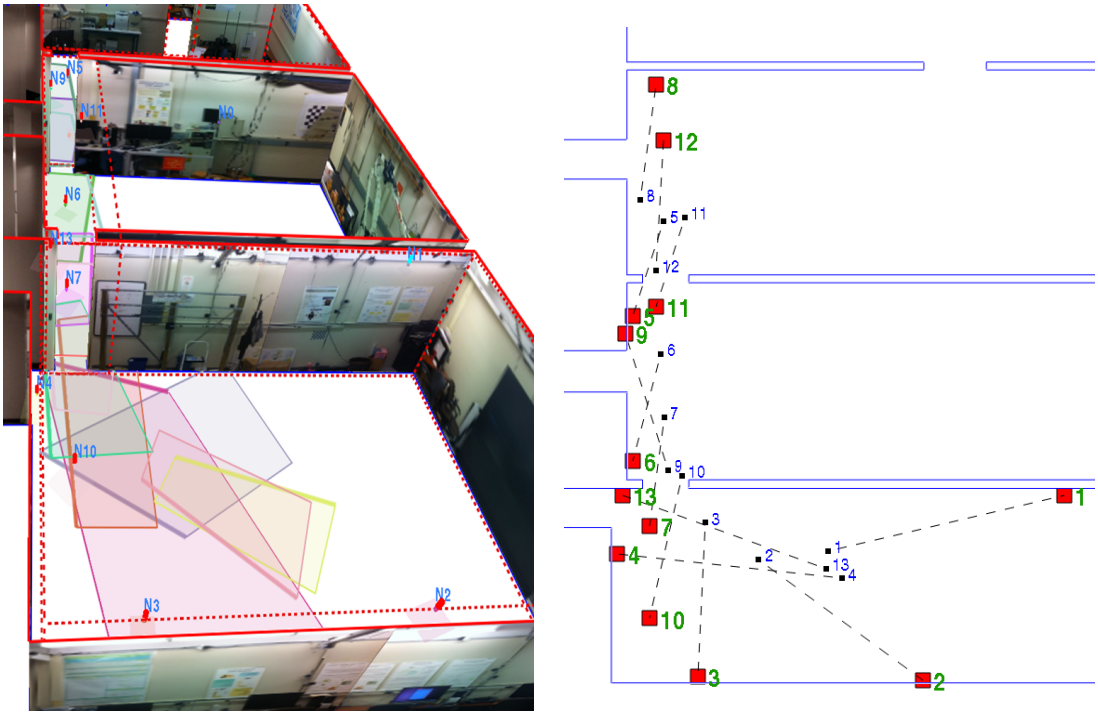


Figure 5.18.: Average tracking errors (ATEs) of 3 - 8 Lab sequences. From left to right, each column shows ATEs for the synchronized, GMM, and Parzen particle filters, respectively. From top to bottom, each row depicts ATEs under loss model 1, 2, and 3, respectively.

5.2.4 Experiments on a Wireless Camera Network Testbed

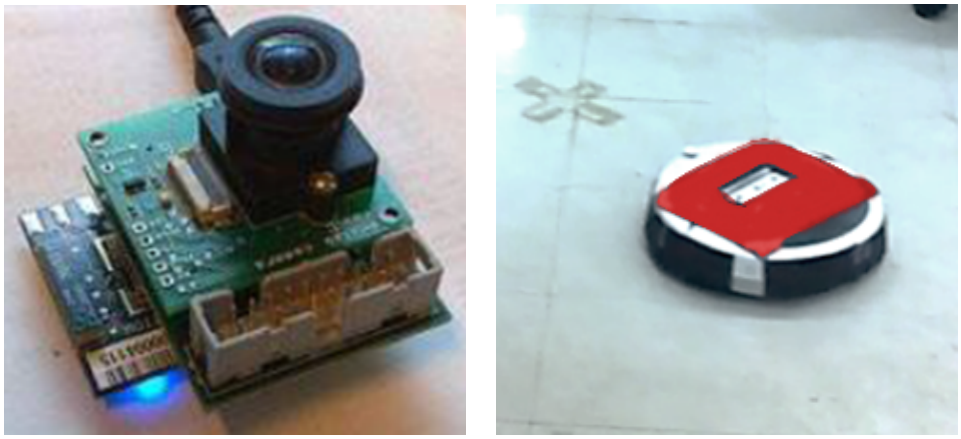
The proposed resource-aware method was evaluated on an Imote2-based real wireless camera network testbed as shown in Fig. 5.19. The Robot Vision Lab (RVL) testbed consists of 13 Imote2 nodes, shown in Fig.5.20 (a), and covers a doorway across three consecutive rooms where each room size is roughly $20ft \times 20ft$ ($6m \times 6m$). The sensing rate of the camera nodes was set to capture a 320 by 240 image at every 1.6 second; the extremely low sensing rate is due to the hardware limitations. The target object (an iRobot Create with color markers), shown in Fig. 5.20 (b), was navigated on a pre-determined track (the ground-truth) using a remote control. We compared the distributed particle filter without the resource-aware capability using 2, 4, and 6 packet loads with the resource-aware particle filter with $M_{max} = 0.2$. The Parzen distributed particle filter, which is robust to packet losses during data aggregation at the cluster head, is applied to all the implementations. The network protocol stack that supports the object tracking application is built with typical wireless sensor network protocols for realistic experimental environments, which includes the energy-efficient random-access MAC protocol T-MAC [55] with 30% duty cycle. We also utilized a dynamic camera clustering protocol [7] to enable mobile object tracking with static wireless cameras.

As we demonstrated in the previous section, all the implementations of the Parzen particle filters achieve similar tracking performance as measured by ATS and ATE. We can observe the same behavior in Fig. 5.21 (a) and (b), which show the ATS and ATE measured in the experiment on the testbed. As shown in Fig. 5.21 (d), as more transmission packets are assigned per node, the contention for medium becomes more severe, resulting in increased packet losses. It is obvious that such packet losses cause unnecessary energy consumption, and more importantly the increased contention could potentially cause the loss of more critical packets, such as the packets



(a) (b)

Figure 5.19.: The RVL wireless camera network testbed. (a) A 3D view of the testbed: The camera sensing range of each camera is shown as a colored polygon. (b) A floorplan view of the testbed: The location of the camera nodes are indicated by the red boxes and the center of each camera sensing range is shown with a small black dot (a camera and its sensing center is connected with dotted lines).



(a) (b)

Figure 5.20.: Network camera and target object. (a) the Imote2 smart camera and (b) the target object.

required for the operation of the clustering algorithm. For example, if the packets used for cluster propagation were dropped as a consequence of the increased contention, more frequent tracking failures would take place. When the resource-aware method is applied, however, the packet loss rate is reduced by controlling the packet generation rate at the cluster members, preventing the medium from being saturated and thus leaving enough bandwidth available for the other protocols such as the clustering protocol. Such performance gain of the proposed resource-aware approach is well-captured by the improved DEE as shown in Fig. 5.21 (c). Fig. 5.22 shows the estimated target trajectories of the non resource-aware particle filter with 6 packet load and the resource-aware particle filter. Note that the tracking errors are caused by both communication failures and calibration inaccuracies.

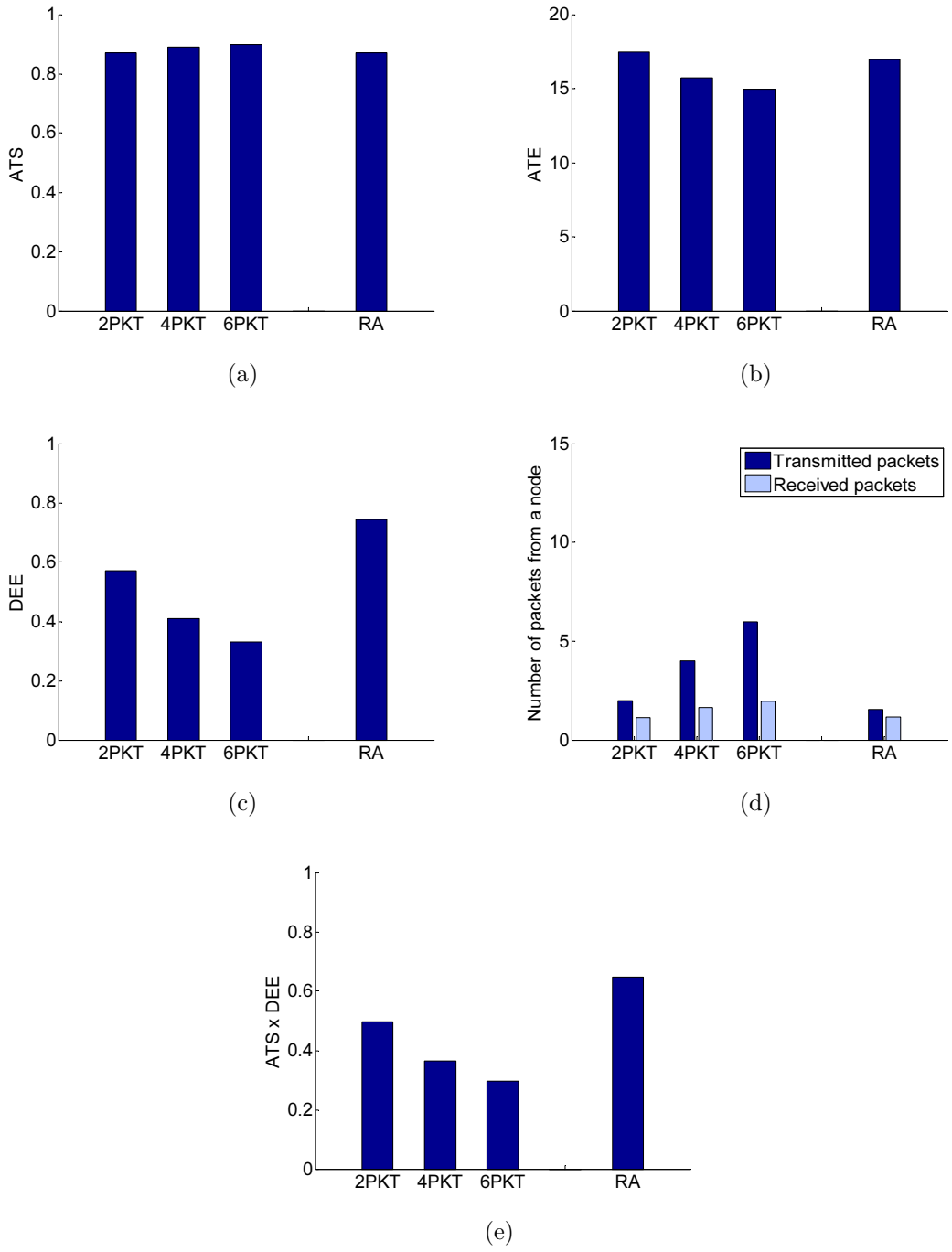
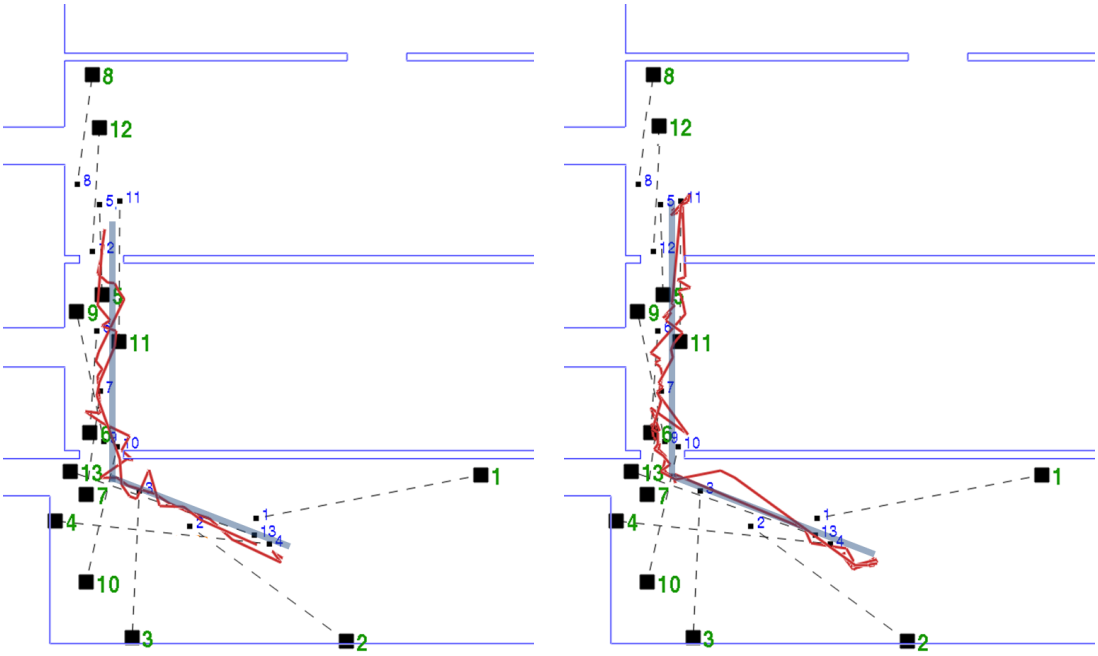


Figure 5.21.: Distributed particle filter performance on the RVL wireless camera network testbed. (a) Average tracking scores (ATSs), (b) average tracking errors (ATEs), (c) delivery energy efficiencies (DEEs), and (d) TX and RX packets. (e) Average tracking scores with delivery energy efficiency (DEE $\times$ATS).



(a) (b)

Figure 5.22.: Object trajectories. The blue lines show the ground-truth of the target trajectories and the red lines indicate the estimated target trajectories; (a) non resource-aware particle filter with 6 packet load and (b) resource-aware particle filter with M_{max}= 0.2.

6. CONCLUSIONS

In this dissertation, we proposed a resource-aware particle filtering scheme for WCNs. The resource-aware method utilizes an optimization process to achieve a certain level of packet loss rate for the communication of tracking information and to preserve packet delivery energy efficiency. This packet allocation procedure alleviates the data loss effects in particle filtering by adjusting the amount of packets generated and transmitted by each camera, thereby reducing the amount of collisions due to a saturated communication medium. This procedure allows collaborative tracking to be carried out using as much data as possible and ultimately leads to more accurate tracking. In addition, we presented three different mechanisms for the exchange of particle information: the synchronized, the GMM, and the Parzen particle filters. We extensively evaluated the performance of these particle filters and analyzed their behaviors under different network traffic conditions. Our experimental results showed that whereas the synchronized particle filter cannot tolerate even small amounts of communication failures, both the GMM and the Parzen particle filters can be employed in lossy environments. The Parzen particle filter is especially robust to communication failures and performs relatively well even when communication quality is extremely low. Our proposed resource-aware packet allocation mechanism improved both the tracking performance and the energy efficiency of the GMM particle filter. As for the Parzen particle filter, because tracking performance is generally robust even in the presence of severe communication problems, the main benefit of the resource-aware approach is to significantly increase its energy efficiency. The resource-aware distributed par-

ticle filters are an important tool for the development of effective WCNs that can be employed in real-world surveillance applications.

As for future work, we will consider (1) a resource-aware approach to compute the optimal number of cameras that should join a cluster in the form of cluster members according to the current network conditions and (2) a resource-allocation methodology to allow weighted packet allocation to individual cluster members given the available resources.

LIST OF REFERENCES

LIST OF REFERENCES

[1] W. Du and J. Piater, "Multi-camera people tracking by collaborative particle filters and principle axis-based intergration," in *Proceedings of Asian Conference on Computer Vision*, 2007.

[2] C. R. del Blanco, R. l Mohedano, N. Garcia, L. Salgado, and F. Jaureguizar, "Color-based 3D particle filtering for robust tracking in heterogeneous environments," in *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.

[3] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. V. Gool, "Color-based object tracking in multi-camera environments," in *Proceedings of the 25th Pattern Recognition Symposium, DAGM*, 2003.

[4] G. Kayumbi and A. Cavallaro, "Robust homography-based trajectory transformation for multi-camera scene analysis," in *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, 2006.

[5] R. l Mohedano, C. R. del Blanco, F. Jaureguizar, L. Salgado, and N. Garcia, "Robust 3d people tracking and positioning system in a semi-overlapped multi-camera environment," in *Proceedings of IEEE International Conference on Image Processing*, 2008.

[6] L. Nachman, J. Huang, J. Shahabdeen, R. Adler, and R. Kling, "IMOTE2: Serious computation at the edge," in *Proceedings of Wireless Communications and Mobile Computing Conference*, pp. 1118–1123, August 2008.

[7] H. Medeiros, J. Park, and A. Kak, "A light-weight event-driven protocol for sensor clustering in wireless camera networks," in *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 203 – 210, September 2007.

[8] H. Medeiros, J. Park, and A. Kak, "Distributed object tracking using a cluster-based Kalman filter in wireless camera networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, pp. 448–463, August 2008.

[9] M. Coates, "Distributed particle filters for sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pp. 99–107, 2004.

[10] X. Sheng and Y.-H. Hu, "Distributed particle filters for wireless sensor network target tracking," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 845–848, March 2005.

[11] Y. Huang, W. Liang, H.-B. Yu, and Y. Xiao, "Target tracking based on a distributed particle filter in underwater sensor network," *Wireless Communications and Mobile Computing*, vol. 8, pp. 1023–1033, October 2008.

[12] P. J. Shin, H. Medeiros, J. Park, and A. Kak, "Predictive duty cycle adaptation for wireless camera networks," in *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras*, pp. 1–6, 2011.

[13] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2000.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[15] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 1150–1157, 1999.

[16] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proceedings of European Conference on Computer Vision*, pp. 661–675, 2002.

[17] M. Isard and A. Blake, "CONDENSATION - conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[18] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "A color-based particle filter," in *Proceedings of International Workshop on Generative-Model-Based Vision*, vol. 2002/01, pp. 53–60, 2002.

[19] G. Bradski, "Computer vision face tracking as a component of a perceptual user interface," in *Proceedings of Workshop on Applications of Computer Vision*, pp. 214–219, 1998.

[20] H. Chen and T. Liu, "Trust-region methods for real-time tracking," in *Proceedings of IEEE International Conference on Computer Vision*, pp. 717–722, 2001.

[21] K. I. Kim, K. Jung, and J. H. Kim, "Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm," *IEEE Transactions on Pattern Analysis And Machine Intelligence*, vol. 25, pp. 1631–1639, December 2003.

[22] G. D. Hager, M. Dewan, and C. V. Stewart, "Multiple kernel tracking with SSD," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 790–797, 2004.

[23] C. Hue, J.-P. L. Cadre, and P. Perez, "Sequential Monte Carlo methods for multiple target tracking and data fusion," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 309–325, 2002.

[24] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[25] A. Doucet, N. D. Freitas, and N. Gordon, *Sequential Monte Carlo methods in practice*. Springer, 2001.

[26] A. Papoulis and S. U. Pillai, *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 4th ed., 2002.

[27] N. Gordon, D. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," in *IEE Proceedings F, Radar and Signal Processing*, vol. 140, pp. 107–113, April 1993.

[28] J. S. Liu and R. Chen, "Sequential monte carlo methods for dynamical systems," *Journal of American Statistical Association*, vol. 93, pp. 1032–1044, 1998.

[29] W. Li, "Camera sensor activation scheme for target tracking in wireless visual sensor networks," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

[30] P. J. Shin, *Predictive Duty Cycling of Radios and Cameras using Augmented Sensing in Wireless Camera Networks*. Ph.D. dissertation, Purdue University, West Lafayette, IN, 2013.

[31] S. Fleck, F. Busch, and W. Strabser, "Adaptive probabilistic tracking embedded in smart cameras for distributed surveillance in a 3D model," *EURASIP Journal on Embedded Systems*, vol. 2007, p. 17 pages, January 2007.

[32] W. Qu, D. Schonofeild, and M. Mohamed, "Distributed Bayesian multiple-target tracking in crowded environments using multiple collaborative cameras," *EURASIP Journal on Advances in Signal Processing*, 2007.

[33] M. Ridley, B. Upcroft, L.-L. Ong, S. Kumar, and S. Sukkarieh, "Decentralised data fusion with Parzen density estimates," in *Proceedings of Intelligent Sensors, Sensor Networks and Information Processing Conference*, pp. 161–166, 2004.

[34] G. Ing and M. Coates, "Parallel particle filters for tracking in wireless sensor networks," in *Proceedings of IEEE Workshop on Signal Processing Advances in Wireless Communications*, 2005.

[35] L.-L. Ong, B. Upcroft, M. Ridley, T. Bailey, S. Sukkarieh, and H. Durrant-Whyte, "Consistent methods for descentralised data fusion using particle filters," in *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 85–91, September 2006.

[36] L.-L. Ong, T. Bailey, H. Durrant-Whyte, and B. Upcroft, "Decentralised particle filtering for multiple target tracking in wireless sensor networks," in *Proceedings of International Conference on Information Fusion*, pp. 1–8, 2008.

[37] L.-L. Ong, B. Upcroft, M. Ridley, T. Bailey, S. Sukkarieh, and H. Durrant-Whyte, "Decentralised data fusion with particles," in *Proceedings of Australasian Conference on Robotics and Automation*, 2005.

[38] L.-L. Ong, B. Upcroft, T. Bailey, M. Ridley, S. Sukkarieh, and H. Durrant-Whyte, "A decentralised particle filter algorithm for multi target tracking across multiple flight vehicles," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4539–4544, 2006.

[39] S. Farahmand, S. I. Roumeliotis, and G. B. Giannakis, "Set-membership constrained particle filter: Distributed adaptation for sensor networks," *IEEE Transactions on Signal Processing*, vol. 59, pp. 4122–4138, September 2011.

[40] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, pp. 181–188, April 2005.

[41] H. Ma and B. W.-H. Ng, "Collaborative data and information processing for target tracking in wireless sensor networks," in *Proceedings of IEEE International Conference on Industrial Informatics*, pp. 647–652, August 2006.

[42] L. Zuo, K. Mehrotra, P. K. Varshney, and C. K. Mohan, "Bandwidth-efficient target tracking in distributed sensor networks using particle filters," in *Proceedings of International Conference on Information Fusion*, pp. 1–4, July 2006.

[43] D. Gu, "Distributed particle filter for target tracking," in *Proceedings of IEEE International Conference in Robotics and Automation*, pp. 3856–3861, April 2007.

[44] C. Song, H. Zhao, and W. Jing, "Asynchronous distributed PF algorithm for WSN target tracking," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, pp. 1168–1172, June 2009.

[45] W. Gao, H. Zhao, C. Song, and J. Xu, "A new distributed particle filtering for WSN target tracking," in *Proceedings of International Conference on Signal Processing Systems*, pp. 334–337, 2009.

[46] S. Kung, M. Mak, and S. Lin, *Biometric Authentication: A Machine Learning Approach*. Prentice Hall, 2004.

[47] Y. G. Iyer, S. Gandham, and S. Venkatesan, "STCP: A generic transport layer protocol for wireless sensor networks," in *Proceedings of International Conference on Computer Communications and Networks*, pp. 449–454, October 2005.

[48] B. Hull, K. Jamieson, and H. Balakrishnan, "Mitigating congestion in wireless sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp. 134–147, 2004.

[49] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "CODA: Congestion detection and avoidance in sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 266–279, November 2003.

[50] C. Wang, K. Sohraby, V. Lawrence, B. Li, and Y. Hu, "Priority-based congestion control in wireless sensor networks," in *Proceedings of IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, pp. 22–31, 2006.

[51] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall, 2001.

[52] S. Khan, A.-S. K. Pathan, and N. A. Alrajeh, *Wireless Sensor Networks: Current Status and Future Trends*. CRC Press, 2012.

[53] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) spec," 1998.

[54] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.*, 2002.

[55] T. Van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 171–180, 2003.

[56] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 1–13, 2003.

[57] E. K. P. Chong and S. H. Zak, *An Introduction to Optimization.* Wiley-Interscience, 2008.

[58] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* Wiley-Interscience, 2nd ed., 2006.

[59] H. Medeiros, J. Park, and A. Kak, "A parallel color-based particle filter for object tracking," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1–8, 2008.

[60] B. Titzer, D. K. Lee, and J. Palsberg, "Avrora: Scalable sensor network simulation with precise timing," in *Proceedings of International Symposium on Information Processing in Sensor Networks*, pp. 25–27, April 2005.

[61] Ecole Polytechnique Federale de Lausanne, "Campus sequences." [Online]. Available: http://cvlab.epfl.ch/data/pom.

[62] Reading University, "Pets 2009 data sequence." [Online]. Available: http://www.cvg.rdg.ac.uk/PETS2009.

[63] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[64] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B*, vol. 39, pp. 1–38, 1977.

[65] B. A. Turlach, "Bandwidth selection in kernel density estimation: A review," in *CORE and Institut de Statistique*, 1993.

APPENDICES

A. GMM DENSITY ESTIMATION

Let us say that we have K random samples $x_1, x_2, \cdots, x_K$ from a PDF $p(x)$. The PDF can inversely be estimated from the samples through N Gaussian mixture models. The number of mixture models, N, can be interpreted as the number of modes of the estimated PDF. These modes can be obtained from clustering the samples using the k-means clustering algorithm [63] and EM [64] methods. In GMM, each mode is resented by a single Gaussian function whose mean, variance, and weight can be computed by EM. The GMM estimation of the PDF, $\hat{p}(x)$, is then formed as

$$\hat{p}(x) = \frac{1}{Z} \sum_{n=1}^{N} c_n \mathcal{N}(x - m_n, \sigma_n^2), \tag{A.1}$$

where Z is the normalization constant, c_n is the n^{th} weight, and $\mathcal{N}(m, \sigma^2)$ is a Gaussian PDF with mean m and variance σ^2.

B. PARZEN WINDOW DENSITY ESTIMATION

Let us say that we have N random samples $x_1, x_2, \cdots, x_N$ from a PDF $p(x)$. Then, the Parzen's estimation (or kernel estimation) of the PDF, $\hat{p}(x)$, is formed as

$$\hat{p}(x) = \frac{1}{Z} \sum_{n=1}^{N} K_h(x - x_n), \tag{B.1}$$

where Z is the normalization constant and K_h is a Parzen's kernel. The kernel has to be symmetric and positive. The Gaussian function is one of the most commonly used kernels for the Parzen's estimate:

$$K_h(x) = e^{-x^2/h}, \tag{B.2}$$

where h is the kernel bandwidth. Note that the estimated quality mainly depends on the choice of h rather than the choice of kernel [65]. As we can see in Eq. (B.2), the kernel bandwidth controls the kernel width and therefore it is important to choose a proper kernel bandwidth h for the PDF estimation. Intuitively, if h is small, the estimation $\hat{p}(x)$ looks like a discrete approximation which only can describe $p(x)$ on near samples. If h is large, the estimation becomes too smooth to characterize $p(x)$ in detail. Simply, the kernel width h depends on the number of samples. For the Gaussian kernel, the following choice is known to be a good way for assigning the kernel bandwidth [65]:

$$h \propto \sigma^2 N^{-1/5}, \tag{B.3}$$

where σ^2 is the sample variance and N is the number of samples.

In a distributed particle filter implementation, it is possible to consider the support points that have equal weights from the resampling process as IID random samples. This fact allows us to approximate the particle representation of PDFs using the Parzen's kernel estimation.

VITA

VITA

Kihyun Hong is a Ph.D. candidate in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, IN. He received the B.Eng. degree in electronics engineering from Kwangwoon University, Seoul, Korea and the M.S. in electrical engineering from Stanford University, CA. He worked at the Digital Media Research and Development Center, Samsung Electronics Corporation Ltd., Suwon, Korea before joining the Ph.D. program. His research interests are camera networks, computer vision applications, and perceptual video/image signal processing.